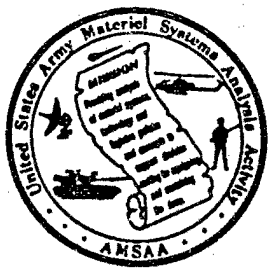
AMSAA



TECHNICAL REPORT NO. TR-687

MATHEMATICA TEMPLATES FOR SEQUENTIAL RELIABILITY/MAINTAINABILITY TEST DESIGN AND ANALYSIS

JUNE 2001

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

U.S. ARMY MATERIEL SYSTEMS ANALYSIS ACTIVITY
ABERDEEN PROVING GROUND, MARYLAND 21005-5071

DISCLAIMER

The findings in this report are not to be construed as an official Department of the Army position unless so specified by other official documentation.

WARNING

Information and data contained in this document are based on the input available at the time of preparation.

TRADE NAMES

The use of trade names in this report does not constitute an official endorsement or approval of the use of such commercial hardware or software. The report may not be cited for purposes of advertisement.

{PRIVATE }REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (LEAVE BLANK)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 2001	Technical Report

4. TITLE AND SUBTITLE
Mathematica Templates for Sequential Reliability/Maintainability Test Design and Analysis

5. FUNDING NUMBERS

6. AUTHOR(S)
Dr. Michael Cushing

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Director
U.S. Army Materiel Systems Analysis Activity
392 Hopkins Road
Aberdeen Proving Ground, MD 21005-5071

8. PERFORMING ORGANIZATION REPORT NUMBER
TR-687

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Director
U.S. Army Materiel Systems Analysis Activity
392 Hopkins Road
Aberdeen Proving Ground, MD 21005-5071

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (Maximum 200 words)
This report documents the recent design and/or analysis of sequential reliability test plans for three Army systems. Also included in this report is a new, simulation-based method for designing a hypergeometric test plan for acceptance of maintenance troubleshooting procedures based on sequential sampling. The design and analysis of sequential test plans is computationally challenging thus implementation in software is essential. For this reason, each chapter and appendix of this report was written in *Mathematica*, a leading commercial mathematics software. The test designs and analyses contained within this report constitute a basic set of templates for sequential test planning in the future.

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT

THIS PAGE INTENTIONALLY LEFT BLANK.

Acknowledgement

The US Army Materiel Systems Analysis Activity recognizes the following individuals for contributing to this report:

John A. Sereno

West Point Cadet George Bissias

The author expresses his appreciation to Cadet Bissias for implementation of the exact-analysis method of Epstein, Patterson and Qualls (1963) in *Mathematica*. The author also expresses his appreciation to Mr. Sereno for development of the exponential-sequential test design package (Appendix D).

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

Until quite recently, the Army (and many other customers for that matter) selected and contractually mandated specific, statistical reliability test plans. With the advent of acquisition reform, the contractor often selects or designs test plans and is encouraged to be innovative in doing so. For this reason, sequential test plans, rarely used in recent years, have gained favor since they offer markedly reduced test lengths compared with fixed-length plans. The design and analysis of sequential test plans is computationally challenging. During the past year or so, urgent requirements arose on a number of Army programs for timely analysis of proposed sequential test designs as well as for the design of new or alternative plans. It became apparent that new methods, and the implementation of both new and established methods in commercial mathematics software, was urgently needed. The purpose of this report is to disseminate recent progress in this area.

This report documents the design and/or analysis of sequential test plans for four Army systems. The first three case studies illustrate the application of these methods to reliability qualification testing. Important benefits to each program were realized. The fourth case study illustrates a new, simulation-based method for designing a hypergeometric test plan for acceptance of maintenance troubleshooting procedures based on sequential sampling.

A key accomplishment included in this report concerns the exact-analysis method for exponential sequential test designs. Previously, such exact-analysis methodology was, for all practical purposes, restricted to the statistical research community. Indeed, little practical use was found for these methods during the past forty years. It was possible to re-formulate and implement the exact-analysis method in modern mathematics software in a form that, for the first time, can be routinely used by test planners. It was deemed decisively advantageous to undertake this effort because of the resurgence of truncated exponential sequential test designs, the properties of which are very difficult to obtain otherwise.

The test designs and analyses contained within this report constitute a basic set of electronic templates for sequential test planning in the future. The electronic form of each chapter and appendix of this report is a *Mathematica* 4 notebook. All of the methodology, computations and graphics in this report are *Mathematica* executables. The results were generated and inserted by *Mathematica*. Thus the technical content of this report is "live" in the sense that it can be re-executed as desired by readers working with the electronic version (provided they have a copy of *Mathematica* 4). Please refer to *The*

Mathematica Book [Wolfram 1999] for information on this software. Additional information, including a free reader, is available at *http://www.wolfram.com/*.

Chapter 2

Analysis and Confirmation of a Proposed Exponential Sequential Test Plan: Case Study 1

Introduction

An Army imaging system was to be subjected to a fixed-configuration reliability demonstration test. The required Mean Time Between Failures (MTBF) was 503 hours and the consumer risk (i.e., the worst-case risk of the Army accepting the system if the true MTBF is lower than 503 hours) was not to appreciably exceed 20%. The contractor proposed the following exponential sequential test design:

- lower-test MTBF = 503 hours
- upper-test MTBF = 2*503 = 1006 hours
- consumer risk = producer risk = 20%
- decision rules given by the following table:

Failures	Reject Time (hours) ≤	Accept Time (hours) ≥
0	N/A	1395
1	N/A	2092
2	N/A	2789
3	697	3487
4	1395	4184
5	2092	4881
6	2789	5578
7	5578	N/A

The Army test planners and evaluators needed to quickly analyze the proposed test plan, verify that the consumer risk met requirements and calculate the properties to include obtaining the operational-characteristic and expected test time curves.

We analyzed the proposed decision rules using two methods and the results were compared. The rules were first simulated and then an exact analysis was performed. The simulation functions and results are

provided in this chapter as are key results of the exact analysis. The exact-analysis details are provided in Appendix A.

Simulation

The decision rules are simulated for two values of the true MTBF. The true MTBF is first assumed to be equal to the lower-test MTBF of 503 hours and is then assumed to be equal to the upper-test MTBF of 1,006 hours. Each simulation produces an approximate value for the acceptance probability, expected quantity of failures and expected test time. This provides approximate values for the consumer and producer risks since the former is defined as the acceptance probability if the true MTBF equals the lower-test MTBF and the latter can be defined as one minus the acceptance probability when the true MTBF equals the upper-test MTBF.

It should be noted that many of the executable cells in this chapter have been designated as initialization cells. As a result, all of the simulations (except for the timing experiments) can be executed by directing the kernel to evaluate the initialization cells.

- **Define and Plot the Decision Rules**

First let's define and plot the proposed rules for arriving at accept and reject decisions. The rules for arriving at an accept decision provided in the introduction of this chapter are (hours):

```
accept[0] = 1395;

accept[1] = 2092;

accept[2] = 2789;

accept[3] = 3487;

accept[4] = 4184;

accept[5] = 4881;

accept[6] = 5578;
```

The rules for arriving at a reject decision provided are (hours):

```
reject[3] = 697;

reject[4] = 1395;

reject[5] = 2092;

reject[6] = 2789;

reject[7] = 5578;
```

Plotting the decision rules helps one visualize them. We will use the function `MultipleListPlot` which is defined in the standard add-on package `Graphics`MultipleListPlot`. This package must be loaded first.

```
Needs["Graphics`MultipleListPlot`"]
```

In order to make the decision rules conform to the syntax requirements of `MultipleListPlot`, we will generate a list of time-failure pairs for first the accept rules and then the reject rules.

```
acceptpoints = Table[{accept[i], i}, {i, 0, 6}]
```

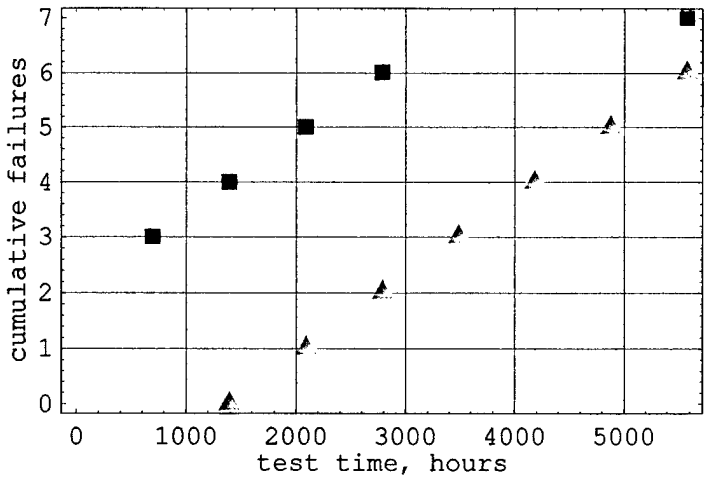
```
{{1395, 0}, {2092, 1}, {2789, 2},
 {3487, 3}, {4184, 4}, {4881, 5}, {5578, 6}}
```

```
rejectpoints = Table[{reject[i], i}, {i, 3, 7}]
```

```
{{697, 3}, {1395, 4}, {2092, 5}, {2789, 6}, {5578, 7}}
```

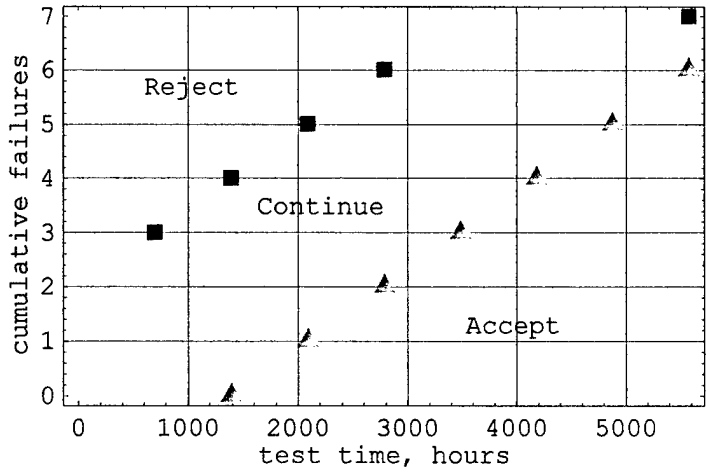
Now we can plot the rules using triangles for the accept points and boxes for the reject points.

```
MultipleListPlot[acceptpoints, rejectpoints,
  SymbolShape → {PlotSymbol[Triangle, 5], PlotSymbol[Box, 3]},
  SymbolStyle → {RGBColor[0, 1, 0], RGBColor[1, 0, 0]}, Frame → True,
  FrameLabel → {"test time, hours", "cumulative failures"},
  GridLines → Automatic];
```



Identification of the reject, continue and accept regions can be overlaid thus:

```
Show[%, Graphics[{Text["Reject", Scaled[{0.2, 0.8}]], Text["Continue",
    Scaled[{0.4, 0.5}]], Text["Accept", Scaled[{0.7, 0.2}]]}]];
```



- **Define Simulation Function**

In this section we will define a function which simulates exponential sequential tests. Since the function uses random-number generation defined in the standard add-on package `Statistics`ContinuousDistributions``, we must first load this package.

```
Needs["Statistics`ContinuousDistributions`"]
```

Now we define the new function ExponentialSequentialSimulation.

```
ExponentialSequentialSimulation[trueMTBF_?Positive,
  acceptfun_Symbol, rejfun_Symbol, trials_Integer?Positive] :=
 Module[{cumtermtime = 0, cumfail = 0, cumaccept = 0, i, testtime},
  Do[i = 0; testtime = 0; While[reject[i] < testtime < accept[i - 1],
    testtime += Random[ExponentialDistribution[1/trueMTBF]]; i++];
   If[testtime ≥ accept[i - 1], cumtermtime += accept[i - 1];
    cumfail += i - 1; cumaccept++, cumtermtime += testtime; cumfail += i],
   {trials}]; {AverageTerminationTime → cumtermtime/trials,
   AverageFailureQuantity → cumfail/trials,
   AverageAcceptFraction → cumaccept/trials}]
```

ExponentialSequentialSimulation provides an approximate average for the fraction of tests that result in an accept decision. The fraction of tests that result in a reject decision may be calculated by subtracting this result from one. ExponentialSequentialSimulation also provides average values for the unconditional test-termination time and failure quantity.

It should be noted that a different approach to exponential sequential simulation is taken in Appendix B. The approach taken in this chapter is more efficient in terms of execution time and memory usage, thus it permits one to perform larger simulations. The approach taken in Appendix B is easier to setup and saves more simulation data at the expense of additional execution time and memory.

It should also be noted that the function in the standard add-on package Statistics`Continuous-Distributions` for generating machine-precision, pseudorandom numbers from the exponential distribution is used here but not in Appendix B. In Appendix B, arbitrary-precision pseudorandom numbers are generated in order to obtain highly-accurate results as recommended by McCullough (2000), the penalty for which is increased execution time.

- **Identify and Define Additional Rules**

In order for the simulation function `ExponentialSequentialSimulation` to run correctly, there must be an appropriate accept and reject time at any potential quantity of failures that may be encountered during the simulation.

First let's consider the accept rules. An accept time is needed for the i - 1 failure where i ranges from 0 to the maximum quantity of failures allowed by the rules. The maximum quantity of failures allowed by subject test plan is 7. A review of the accept rules already defined reveals that the only additional rule we need is when i equals 0. This corresponds to a failure quantity of -1, a physically impossibility. This is a computational precondition for the simulation to begin. An accept time of 1 for the failure quantity of -1 will work for any test plan.

```
accept[-1] = 1;
```

An additional rule would have been needed if any accept rules were missing for failure quantities greater than or equal to zero. There must be an accept time for any physically possible quantity of failures. If there are accept rules missing, the next rule should be used. For example, if there's no failure-free accept time, the failure-free and one-failure accept times should be equated.

Now let's consider the reject rules. A reject time is needed for the i th failure where i ranges from 0 to the maximum quantity of failures allowed by the rules. The maximum quantity of failures allowed by subject test plan is 7. A review of the reject rules already defined reveals that additional rules are needed when i equals 0, 1 and 2. This is not unusual. With many exponential sequential test designs, there may not be a way to reject with a small quantity of failures. It is computationally convenient to assign a time of -1 for these cases.

```
reject[0] = -1;

reject[1] = -1;

reject[2] = -1;
```

- **Simulation Timing Experiment**

It would be wise to determine how long a simulation will take with the accept and reject rules defined herein. This will help us determine how large a simulation is practical on a given computer. We will use a value midway between the lower- and upper-test MTBFs as the assumed true MTBF. The simulation execution time should increase linearly with the quantity of trials. First let's time a 10,000 trial simulation.

```
Timing[
 ExponentialSequentialSimulation[1.5*503, accept, reject, 10000]]
```

$$\{1.93\ \text{Second},\ \{\text{AverageTerminationTime} \to 2753.86,\ \text{AverageFailureQuantity} \to \frac{2267}{625},\ \text{AverageAcceptFraction} \to \frac{5393}{10000}\}\}$$

This took approximately 2 seconds on a computer with a 1.2 GHz Athalon processor and 128MB of RAM. Let's try 100,000 trials next.

```
Timing[
 ExponentialSequentialSimulation[1.5*503, accept, reject, 100000]]
```

$$\{19.12\ \text{Second},\ \{\text{AverageTerminationTime} \to 2751.59,\ \text{AverageFailureQuantity} \to \frac{72743}{20000},\ \text{AverageAcceptFraction} \to \frac{53559}{100000}\}\}$$

This took approximately 20 seconds. A simulation of 1,000,000 trials should require approximately 200 seconds. Let's check.

```
Timing[
 ExponentialSequentialSimulation[1.5*503, accept, reject, 1000000]]
```

$$\{196.46\ \text{Second},\ \{\text{AverageTerminationTime} \to 2753.95,\ \text{AverageFailureQuantity} \to \frac{3651977}{1000000},\ \text{AverageAcceptFraction} \to \frac{532747}{1000000}\}\}$$

The execution time was as predicted.

- **Simulation When True MTBF Equals Lower-Test MTBF**

In this section we will run four simulations assuming that the true MTBF equals the lower-test MTBF of 503 hours. Each will simulate 1,000,000 exponential sequential tests using the rules defined earlier. The results of the 1,000,000-trial simulations are assigned as the value of the symbols *lowertestsim1*, *lowertestsim2*, *lowertestsim3* and *lowertestsim4*.

```
N[lowertestsim1 =
  ExponentialSequentialSimulation[503, accept, reject, 1000000]]

{AverageTerminationTime → 2260.46,
 AverageFailureQuantity → 4.49458, AverageAcceptFraction → 0.190786}


N[lowertestsim2 =
  ExponentialSequentialSimulation[503, accept, reject, 1000000]]

{AverageTerminationTime → 2261.22,
 AverageFailureQuantity → 4.4934, AverageAcceptFraction → 0.191338}


N[lowertestsim3 =
  ExponentialSequentialSimulation[503, accept, reject, 1000000]]

{AverageTerminationTime → 2262.59,
 AverageFailureQuantity → 4.4951, AverageAcceptFraction → 0.191233}


N[lowertestsim4 =
  ExponentialSequentialSimulation[503, accept, reject, 1000000]]

{AverageTerminationTime → 2261.58,
 AverageFailureQuantity → 4.4966, AverageAcceptFraction → 0.190735}
```

The consumer-risk values, sorted from smallest to largest, are:

```
N[Sort[AverageAcceptFraction /.
   {lowertestsim1, lowertestsim2, lowertestsim3, lowertestsim4}]]

{0.190735, 0.190786, 0.191233, 0.191338}
```

With an average of:

$$\frac{\text{Apply[Plus, \%]}}{\text{Length[\%]}}$$

```
0.191023
```

This is close to the desired consumer risk of 20%.

The values for expected test time, sorted from smallest to largest, are:

```
N[Sort[AverageTerminationTime /.
   {lowertestsim1, lowertestsim2, lowertestsim3, lowertestsim4}]]
```

```
{2260.46, 2261.22, 2261.58, 2262.59}
```

With an average of:

```
Apply[Plus, %]
---------------
  Length[%]
```

```
2261.46
```

The values for quantity of failures, sorted from smallest to largest, are:

```
N[Sort[AverageFailureQuantity /.
   {lowertestsim1, lowertestsim2, lowertestsim3, lowertestsim4}]]
```

```
{4.4934, 4.49458, 4.4951, 4.4966}
```

With an average of:

```
Apply[Plus, %]
---------------
  Length[%]
```

```
4.49492
```

- **Simulation When True MTBF Equals Upper-Test MTBF**

In this section we will run four simulations assuming that the true MTBF equals the upper-test MTBF of 503*2 = 1,006 hours. Each will simulate 1,000,000 exponential sequential tests using the rules defined earlier. The results of the 1,000,000-trial simulations are assigned as the value of the symbols *uppertestsim1*, *uppertestsim2*, *uppertestsim3* and *uppertestsim4*.

```
N[uppertestsim1 =
  ExponentialSequentialSimulation[1006, accept, reject, 1000000]]
```

```
{AverageTerminationTime → 2687.38,
 AverageFailureQuantity → 2.67365, AverageAcceptFraction → 0.763462}
```

```
N[uppertestsim2 =
  ExponentialSequentialSimulation[1006, accept, reject, 1000000]]

{AverageTerminationTime → 2688.54,
 AverageFailureQuantity → 2.67251, AverageAcceptFraction → 0.764113}


N[uppertestsim3 =
  ExponentialSequentialSimulation[1006, accept, reject, 1000000]]

{AverageTerminationTime → 2685.47,
 AverageFailureQuantity → 2.67121, AverageAcceptFraction → 0.763396}


N[uppertestsim4 =
  ExponentialSequentialSimulation[1006, accept, reject, 1000000]]

{AverageTerminationTime → 2686.77,
 AverageFailureQuantity → 2.6692, AverageAcceptFraction → 0.764277}
```

The producer-risk values, sorted from smallest to largest, are:

```
N[Sort[1 - AverageAcceptFraction /.
   {uppertestsim1, uppertestsim2, uppertestsim3, uppertestsim4}]]

{0.235723, 0.235887, 0.236538, 0.236604}
```

With an average of:

$$\frac{\text{Apply[Plus, \%]}}{\text{Length[\%]}}$$

```
0.236188
```

This is fairly close to the desired 20% producer risk. One should realize that it is not possible to design a truncated, exponential sequential test that will provide exactly the consumer and producer risks desired. (It's possible but quite difficult to do this in the untruncated case.)

The values for expected test time, sorted from smallest to largest, are:

```
N[Sort[AverageTerminationTime /.
   {uppertestsim1, uppertestsim2, uppertestsim3, uppertestsim4}]]

{2685.47, 2686.77, 2687.38, 2688.54}
```

With an average of:

```
Apply[Plus, %]
--------------
  Length[%]
```

```
2687.04
```

The values for quantity of failures, sorted from smallest to largest, are:

```
N[Sort[AverageFailureQuantity /.
   {uppertestsim1, uppertestsim2, uppertestsim3, uppertestsim4}]]
```

```
{2.6692, 2.67121, 2.67251, 2.67365}
```

With an average of:

```
Apply[Plus, %]
--------------
  Length[%]
```

```
2.67164
```

Key Results from and Comparison with Exact Analysis

An exact analysis was performed and may be found in Appendix A. Key results are included in this section for discussion and comparison.

The stage-by-stage acceptance, continuation and rejection probabilities assuming the true MTBF equals the lower-test MTBF are:

ltMTBFtable

	Time	Σ Accept Pr.	Σ Continue Pr.	Σ Reject Pr.
1.	697.	0.	0.83695	0.16305
2.	1395.	0.06245	0.60771	0.32984
3.	2092.	0.10578	0.43784	0.45638
4.	2789.	0.13583	0.31508	0.5491
5.	3487.	0.15705	0.22654	0.6164
6.	4184.	0.17223	0.10829	0.71948
7.	4881.	0.18313	0.03129	0.78559
8.	5578.	0.19095	0.	0.80905

Each row in the table above sums to one as it should. The acceptance probability at the last stage (i.e., the consumer risk) is approximately 19.10%. This is in agreement with the consumer-risk value of 19.10% simulated earlier in this chapter. A larger simulation would provide agreement with the exact analysis to additional decimal places, if desired. These results essentially validate the claim that this is a 20%-consumer risk test design.

The stage-by-stage acceptance, continuation and rejection probabilities assuming the true MTBF equals the upper-test MTBF are:

utMTBFtable

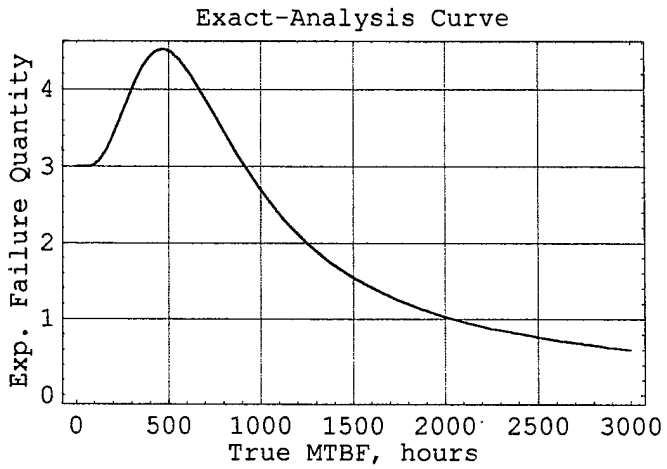
	Time	Σ Accept Pr.	Σ Continue Pr.	Σ Reject Pr.
1.	697.	0.	0.96672	0.03328
2.	1395.	0.2499	0.68401	0.06609
3.	2092.	0.42322	0.48587	0.09091
4.	2789.	0.54339	0.34754	0.10908
5.	3487.	0.62834	0.2494	0.12227
6.	4184.	0.68907	0.15185	0.15908
7.	4881.	0.73263	0.06257	0.2048
8.	5578.	0.76392	0.	0.23608

The rejection probability at the last stage (i.e.,the producer risk) is approximately 23.61%. This is very close to the producer-risk value of 23.62% simulated earlier in this chapter. These results essentially

validate the claim that this is a 20%-producer risk test design.

The expected quantity of failures plot as a function of true MTBF is:

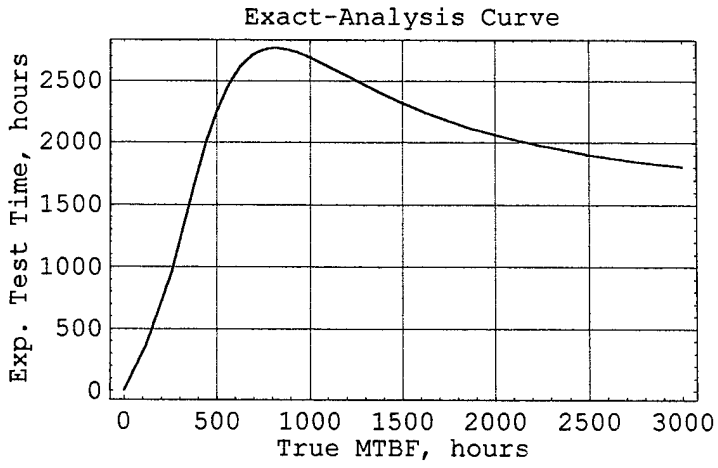
```
Show[expectedfailuresPlot];
```



The expected quantity of failures if the true MTBF equals the lower-test MTBF is 4.49 which agrees with the simulated value of 4.49. The expected quantity of failures if the true MTBF equals the upper-test MTBF is 2.67 which agrees with the simulated value of 2.67.

The plot of expected quantity of test time as a function of true MTBF is:

```
Show[expectedtesttimePlot];
```

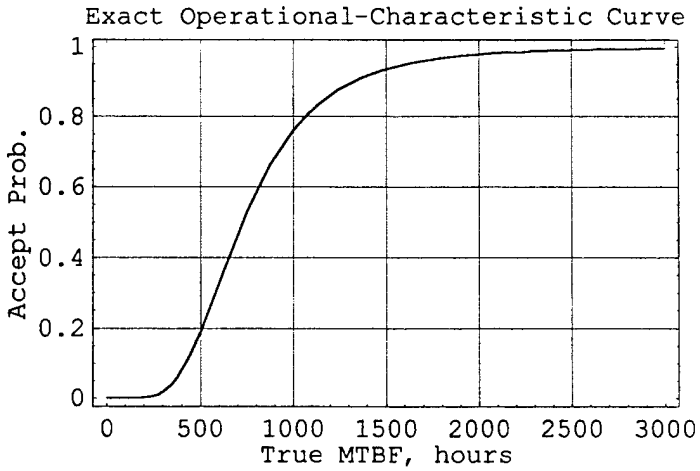


The expected test time if the true MTBF equals the lower-test MTBF is 2,261 hours which agrees with the simulated value of 2,261. The expected test time if the true MTBF equals the upper-test MTBF is

2,686 hours which agrees with the simulated value of 2,687.

The operational-characteristic curve (i.e., the acceptance probability as a function of true MTBF) from appendix A is:

```
Show[ocPlot];
```



The exact analysis was highly beneficial since it provided the operational-characteristic curve, an important test-planning graph. Otherwise, simulations would have been needed at many more than two points in order to characterize the curve.

Summary

This chapter contains simulations of the outcomes of exponential sequential tests which use a proposed collection of exponential sequential decision rules. Two values of the true MTBF are considered. The true MTBF is first assumed to be equal to the lower-test MTBF and then is assumed to be equal to the upper-test MTBF. Based upon 4,000,000 trials, the consumer risk is approximately 19.1%. This is consistent with the assertion that the plan was designed to provide a consumer risk of 20%. Based upon 4,000,000 trials, the producer risk is approximately 23.6%. This is consistent with the assertion that the test plan was designed to provide a producer risk of 20%. This is as close as one can usually get when designing truncated exponential sequential tests. Approximate values were also produced for the expected quantity of failures and expected test time. In all cases, the simulation results are in close agreement with the results of the exact analysis documented in appendix A. The test planners and evaluators were advised that the proposed test plan was as advertised and were provided with the key graphs and tables.

This chapter can serve as a template for the verification of truncated exponential sequential test plans. Indeed, the author has already had occasion to do so many times.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Reliability Test Design Mistake that can Result in High Consumer Risk: Case Study 2

Introduction

The case study in this chapter is based on a recent, contractor-proposed exponential sequential test design that contained a critical, but not infrequent mistake. This chapter was prepared in order to clearly illustrate this mistake and thereby help test designers avoid it in the future.

Test Design

Let's assume that the one-parameter exponential distribution satisfactorily models the time-to-failure of a product. Let us further suppose that we need to design a test with a lower-test value for θ, the exponential distribution parameter, and a not-to-exceed value for consumer risk. We'll choose a lower-test θ of 1480 hours and a consumer risk of 20% in order to work through a concrete example. This implies that if our test is just barely passed, then our 1480 hour requirement will be demonstrated with at least 100 - 20 = 80% one-sided, statistical confidence. In order to design such a test plan, one might proceed as suggested in [Kececioglu 1993, section 7.10]. We begin by calculating the length of a time-terminated test that will result in the desired lower-confidence limit on θ if no failures occur. We need functions for the χ^2 distribution in order to proceed. Functions for the χ^2 distribution are available in the standard add-on package `Statistics`NormalDistribution`` which we now load:

```
Needs["Statistics`NormalDistribution`"]
```

We can implement [Kececioglu 1993, equation 7.34] as follows:

$$\texttt{chiSquareEqn} = \frac{\theta\,\texttt{Quantile[ChiSquareDistribution[2 r + 2], conf]}}{2}$$

$$\theta\,\texttt{InverseGammaRegularized}\left[\frac{1}{2}\,(2 + 2\,\texttt{r}),\ 0,\ \texttt{conf}\right]$$

where r is the quantity of failures and 100*$conf$ is the desired, one-sided statistical confidence level. The function `Quantile` is used in order to obtain percentage points for a distribution. Assuming that the required θ equals 1480 hours, then for the case where r equals 0 and $conf$ equals 0.8 we have:

```
chiSquareEqn /. {θ → 1480, r → 0, conf → 0.8}

2381.97
```

We can continue this process and calculate the test times (rounded off to the closest integer) that correspond to failure quantities up through five and table our results:

```
TableForm[chiSquareTbl =
  Table[{Round[chiSquareEqn /. {conf → 0.8, θ → 1480}], r}, {r, 0, 5}],
 TableHeadings → {None, {"Time", "Failures"}}]
```

Time	Failures
2382	0
4432	1
6333	2
8162	3
9947	4
11701	5

Perhaps it would be correct to interpret the table above as a sequence of decision rules to use in a single test plan as follows:

- accept at 2382 hours if 0 failures have occurred,
- accept at 4432 hours if 1 failure has occurred,
- accept at 6333 hours if 2 failures have occurred,
- accept at 8162 hours if 3 failures have occurred,
- accept at 9947 hours if 4 failures have occurred,
- accept at 11701 hours if 5 failures have occurred and
- reject if 6 failures occur before 11701 hours are accumulated.

Let's analyze these decision rules and determine whether they satisfy our consumer-risk requirement.

Exact Analysis

An exact method was developed by Epstein, Patterson and Qualls [1963] for analyzing a sequence of decision rules such as the one obtained in the previous section. Regardless of how the rules were obtained, they constitute an exponential sequential test plan. This section contains an analysis of the exact stage-by-stage acceptance, continuation and rejection probabilities resulting from the sequence of decision rules. Included are the important special cases that arise at the last stage: consumer risk and operational-characteristic curve. *Mathematica* symbolics are used to obtain results with the parameter θ held symbolic until a numerical value is supplied. The stage-by-stage calculations are performed in such a way that numerical errors that would otherwise accumulate are entirely avoided. The results of all calculations are "exact" but include occurrences of the exponential function. Numerical approximations to any desired precision are provided as well.

Functions contained in the standard add-on package `Statistics`DiscreteDistributions`` are needed by this method which we load now:

```
Needs["Statistics`DiscreteDistributions`"]
```

■ Formulate Reliability Test Plan Decision Rules

In order to apply the exact-analysis method, we need to construct a list of accept points from these decision rules. Each pair will be of the form $\{t_i, i\}$ where the first pair defines the zero-failure accept time, the second pair defines the one-failure accept time, etc. Fortunately, *chiSquareTbl* is structured in exactly this form so we will simply assign it as the value of *accept*.

```
accept = chiSquareTbl
```

```
{{2382, 0}, {4432, 1}, {6333, 2}, {8162, 3}, {9947, 4}, {11701, 5}}
```

We need to construct a list of reject points from these decision rules. Each pair will be of the form $\{t_i, i\}$ where the first pair defines the shortest reject time and the corresponding quantity of failures, the second defines the second-shortest reject time and the corresponding quantity of failures, etc. We can obtain a list of reject points and assign them as the value of the *reject* as follows:

```
reject = chiSquareTbl /. {t_Integer, f_Integer} → {t, 6}
```

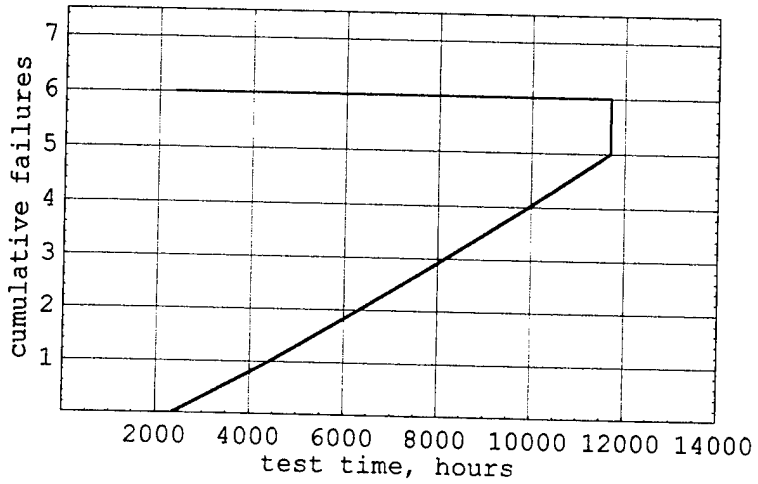
```
{{2382, 6}, {4432, 6}, {6333, 6}, {8162, 6}, {9947, 6}, {11701, 6}}
```

It would be helpful to graphically depict the decision rules for this test design. We will need functions contained in the standard add-on package `Graphics`MultipleListPlot`` which we load now:

```
Needs["Graphics`MultipleListPlot`"]
```

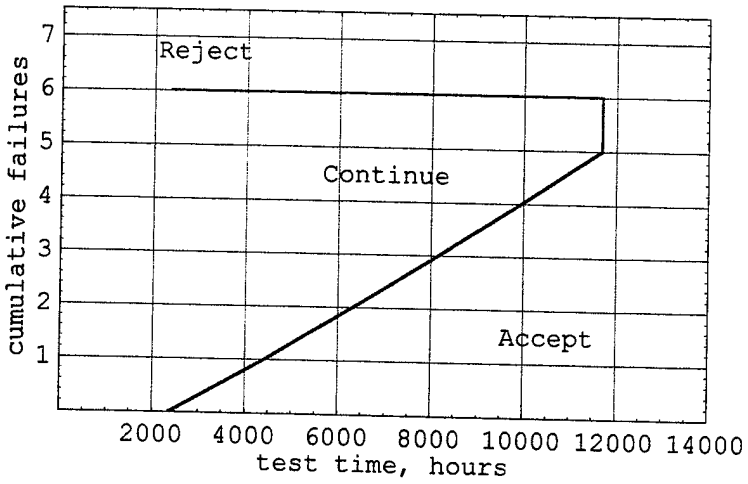
The decision rules are plotted as follows:

```
ListPlot[Join[accept, Reverse[reject]], PlotJoined → True,
  PlotRange → {{0, 14000}, {0, 7.5}}, Frame → True,
  FrameLabel → {"test time, hours", "cumulative failures"}, GridLines →
   Automatic, PlotStyle → {Thickness[0.005], RGBColor[0, 0, 1]}];
```



We've obtained a typical exponential sequential plot. Identification of the reject, continue and accept regions can be overlaid thus:

```
decisionPlot = Show[%,
   Graphics[{Text["Reject", Scaled[{0.22, 0.90}]], Text["Continue",
      Scaled[{0.5, 0.6}]], Text["Accept", Scaled[{0.75, 0.2}]]}]];
```



- **Define Function for Stage Times**

In this step, we will construct a stage-time function. First, a list is needed of the times for each stage. The stage times are comprised of the accept and reject times joined into a single list and sorted from shortest to longest. The list of stage times is constructed as follows:

```
timeValues =
 Sort[Union[First[Transpose[accept]], First[Transpose[reject]]], Less]
```

```
{2382, 4432, 6333, 8162, 9947, 11701}
```

It should be noted that the times are expressed as exact numbers (i.e., either as integers or rational numbers) in order to avoid approximations until after the stage-by-stage calculations are complete. If the times are expressed in decimal form, *Mathematica* will treat them as approximate and will use machine-precision (unless many zeroes are used).

It should also be noted that the function `Union` was used to eliminate any repeats occurring as the two lists were combined.

The quantity of stages is:

```
Length[timeValues]
```

```
6
```

A function which will provide time values as a function of stage, except for the special case of stage zero, is:

```
t[stage_Integer /; stage > 0] := timeValues[[stage]]
```

The initial condition for time [Epstein, et al. 1963, equation 16]:

```
t[i_ /; i == 0] := 0
```

- **Construct Accept-Number Function**

In this step, we will construct an accept-number function. First, we will generate an `InterpolatingFunction` object from *accept*:

```
fA = Interpolation[accept, InterpolationOrder → 1]
```

```
InterpolatingFunction[{{2382, 11701}}, <>]
```

Now, we define a function which will provide an integer-valued accept number for each stage using Epstein, et al. 1963, equation 11:

```
a[stage_Integer /; stage > 0] := -1 /; t[stage] < First[First[accept]]
```

```
a[stage_Integer /; stage > 0] := Floor[fA[t[stage]]]
```

A special case of the accept-number function is defined for the initial condition at stage zero [Epstein, et al. 1963, equation 16]:

```
a[stage_Integer /; stage == 0] := -1
```

- **Construct Reject-Number Function**

In this step, we will construct a reject-number function. First, we will generate an `InterpolatingFunction` object from *reject*:

```
fR = Interpolation[reject, InterpolationOrder → 1]
```

```
InterpolatingFunction[{{2382, 11701}}, <>]
```

Now, we define an function which will provide an integer-valued reject number for each stage using Epstein, et al. 1963, equation 12:

```
r[stage_Integer /; stage > 0] := Ceiling[fR[t[stage]]]
```

A special case of the reject-number function is defined for the initial condition at stage zero:

```
r[stage_ /; stage == 0] := 1
```

- **Tabulation of Accept, Continuation and Reject Points**

In this step, we generate a table of accept, continuation and reject numbers. This is done to provide a convenient stage-by-stage listing of the test plan to be analyzed. The table is generated as follows:

```
TableForm[Transpose[{Range[Length[timeValues]],
   Table[N[t[stage]], {stage, 1, Length[timeValues]}],
   Table[a[stage], {stage, 1, Length[timeValues]}],
   Append[Table[a[stage] + 1, {stage, 1, Length[timeValues] - 1}], NA],
   Append[Table[r[stage] - 1, {stage, 1, Length[timeValues] - 1}], NA],
   Table[r[stage], {stage, 1, Length[timeValues]}]}],
 TableHeadings -> {None, {"Stage", "Time", "Accept",
    "Continue (min)", "Continue (max)", "Reject"}},
 TableSpacing → {1, 1.5}, TableAlignments → Center]
```

Stage	Time	Accept	Continue (min)	Continue (max)	Reject
1	2382.	0	1	5	6
2	4432.	1	2	5	6
3	6333.	2	3	5	6
4	8162.	3	4	5	6
5	9947.	4	5	5	6
6	11701.	5	NA	NA	6

- **Construct Function for Acceptance/Continuation Probability for a Quantity of Failures**

In this step, we construct a function for calculating acceptance/continuation probabilities for a quantity of failures [Epstein, et al. 1963, equation 17]:

```
ACProbability[stage_, failure_, trueTheta_] /;
  And[stage > 0, (a[stage - 1] + 1) ≤ failure ≤ r[stage] - 1] :=
 aclist[stage, failure, trueTheta]
```

```
ACProbability[stage_, failure_, trueTheta_] /;
  And[stage > 0, Not[(a[stage - 1] + 1) ≤ failure ≤ r[stage] - 1]] := 0
```

Two initial conditions for this function are also needed [Epstein, et al. 1963, equation 16]:

```
ACProbability[0, 0, trueTheta_] := 1

ACProbability[0, failure_Integer /; failure > 0, trueTheta_] := 0
```

- **Up-front Calculation of Acceptance/Continuation Probabilities**

In order to reduce execution time, stage-by-stage calculations of acceptance and continuation probabilities are developed in this step.

A function for building up the calculations is:

```
aclistfunction[stage_Integer, failure_Integer, trueTheta_] :=
 aclist[stage, failure, trueTheta] =
    Sum[ACProbability[stage - 1, j, trueTheta],
      {j, a[stage-1]+1, failure}]
      PDF[PoissonDistribution[(t[stage] - t[stage - 1])/trueTheta], failure - j]
```

An indexed variable *aclist* is used to build up the acceptance and continuation probabilities.

The acceptance and continuation points for the stages are:

```
Map[aclistfunction[1, #, trueTheta] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 1]]
```

$$\left\{e^{-2382/\text{trueTheta}}, \frac{2382\, e^{-2382/\text{trueTheta}}}{\text{trueTheta}}, \frac{2836962\, e^{-2382/\text{trueTheta}}}{\text{trueTheta}^2}, \frac{2252547828\, e^{-2382/\text{trueTheta}}}{\text{trueTheta}^3}, \frac{1341392231574\, e^{-2382/\text{trueTheta}}}{\text{trueTheta}^4}, \frac{3195196295609268\, e^{-2382/\text{trueTheta}}}{5\,\text{trueTheta}^5}\right\}$$

```
Map[aclistfunction[2, #, trueTheta] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 2]]
```

$$\left\{\frac{2382\, e^{-4432/\text{trueTheta}}}{\text{trueTheta}}, \frac{7720062\, e^{-4432/\text{trueTheta}}}{\text{trueTheta}^2}, \frac{13073497428\, e^{-4432/\text{trueTheta}}}{\text{trueTheta}^3}, \frac{15340486306474\, e^{-4432/\text{trueTheta}}}{\text{trueTheta}^4}, \frac{6974189018060526 8\, e^{-4432/\text{trueTheta}}}{5\,\text{trueTheta}^5}\right\}$$

```
Map[aclistfunction[3, #, trueTheta] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 3]]
```

$$\left\{\frac{7720062\, e^{-6333/\text{trueTheta}}}{\text{trueTheta}^2}, \frac{27749335290\, e^{-6333/\text{trueTheta}}}{\text{trueTheta}^3}, \frac{54142588804933\, e^{-6333/\text{trueTheta}}}{\text{trueTheta}^4}, \frac{377862055671928093\, e^{-6333/\text{trueTheta}}}{5\,\text{trueTheta}^5}\right\}$$

```
Map[aclistfunction[4, #, trueTheta] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 4]]
```

$$\left\{\frac{27749335290\, e^{-8162/\text{trueTheta}}}{\text{trueTheta}^3}, \frac{104896123050343\, e^{-8162/\text{trueTheta}}}{\text{trueTheta}^4}, \frac{1105066565630177603\, e^{-8162/\text{trueTheta}}}{5\,\text{trueTheta}^5}\right\}$$

```
Map[aclistfunction[5, #, trueTheta] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 5]]
```

$$\left\{\frac{104896123050343\, e^{-9947/\text{trueTheta}}}{\text{trueTheta}^4}, \frac{2041264463854488878\, e^{-9947/\text{trueTheta}}}{5\,\text{trueTheta}^5}\right\}$$

```
Map[aclistfunction[6, #, trueTheta] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 6]]
```

$$\left\{\frac{2041264463854488878\, e^{-11701/\text{trueTheta}}}{5\,\text{trueTheta}^5}\right\}$$

- **Construct Function for Acceptance Probability for a Quantity of Failures**

In this step, we construct a function for calculating acceptance probabilities for a quantity of failures [Epstein, et al. 1963, equation 18]:

```
AcceptanceProbability[stage_Integer, failure_Integer, trueTheta_] :=
 ACProbability[stage, failure, trueTheta]
```

- **Construct and Use Function for Acceptance Probability for Each Stage**

In this step, we construct and use a function for calculating stage-by-stage acceptance probabilities [Epstein, et al. 1963, equation 20]:

```
AcceptanceProbability[stage_Integer, trueTheta_] :=
    a[stage]
     ∑          AcceptanceProbability[stage, failure, trueTheta] /;
failure=a[stage-1]+1
 a[stage - 1] < a[stage]

AcceptanceProbability[stage_Integer, trueTheta_] :=
 0 /; Not[a[stage - 1] < a[stage]]
```

The acceptance probability as a function of the true θ is the sum of the probabilities of acceptance at each stage. This is given by Epstein, et al. 1963, equation 14:

```
AcceptanceProbability[trueTheta_] :=
  n
 ∑ AcceptanceProbability[trueTheta_, i_]
i=1
```

Cumulative Acceptance Probabilities for Each Stage When the True θ is Symbolic

The cumulative acceptance probability for stage one when *trueTheta* is left symbolic is:

```
  1
 ∑  AcceptanceProbability[stage, trueTheta]
stage=1
```

$$e^{-2382/\text{trueTheta}}$$

The result above is exact but partially symbolic. An exact result can be obtained for a specific value of *trueTheta* such as 1480 hours as follows:

```
% /. trueTheta → 1480
```

$$\frac{1}{e^{1191/740}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]

0.1999956906413552797549
```

Requesting a numerical approximation greater than 16 decimal places forces *Mathematica* to perform arbitrary-precision arithmetic rather than rely upon the math co-processor. If the math co-processor is used, *Mathematica* can't guarantee the result. The additional execution time required is negligible for

the calculations in this chapter.

The cumulative acceptance probability for stage two when *trueTheta* is left symbolic is:

$$\sum_{stage=1}^{2} \text{AcceptanceProbability[stage, trueTheta]}$$

$$e^{-2382/\text{trueTheta}} + \frac{2382\, e^{-4432/\text{trueTheta}}}{\text{trueTheta}}$$

The result above is exact but partially symbolic. An exact result can be obtained for a specific value of *trueTheta* such as 1480 hours as follows:

```
% /. trueTheta → 1480
```

$$\frac{1191}{740\, e^{554/185}} + \frac{1}{e^{1191/740}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.2805602681208591005752
```

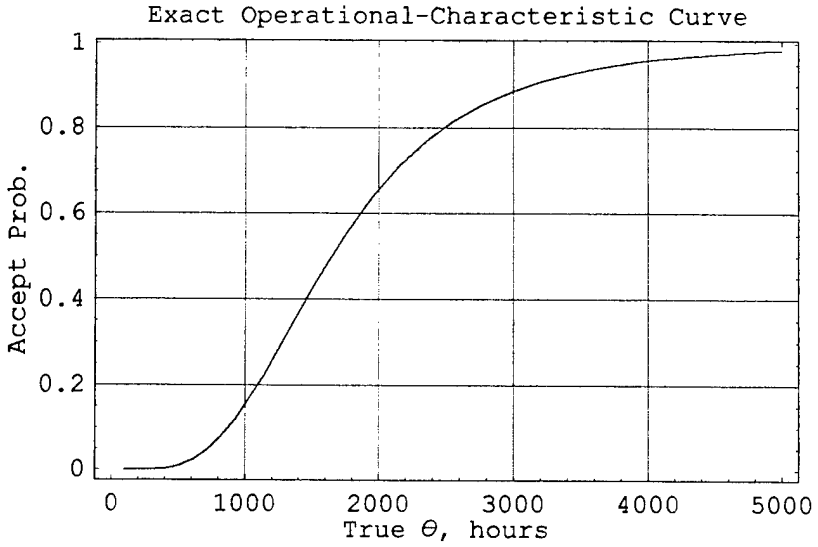
The cumulative acceptance probability for stage six, the final stage, when *trueTheta* is left symbolic will be generated. This is also known as the operational-characteristic function.

$$\text{OCfunction} = \sum_{stage=1}^{6} \text{AcceptanceProbability[stage, trueTheta]}$$

$$e^{-2382/\text{trueTheta}} + \frac{2041264463854488878\, e^{-11701/\text{trueTheta}}}{5\, \text{trueTheta}^5} + \frac{104896123050343\, e^{-9947/\text{trueTheta}}}{\text{trueTheta}^4} + \frac{27749335290\, e^{-8162/\text{trueTheta}}}{\text{trueTheta}^3} + \frac{7720062\, e^{-6333/\text{trueTheta}}}{\text{trueTheta}^2} + \frac{2382\, e^{-4432/\text{trueTheta}}}{\text{trueTheta}}$$

OCfunction provides the exact acceptance probability as a function of *trueTheta*. The exact operational-characteristic curve can now be plotted:

```
Plot[OCfunction, {trueTheta, 100, 5000}, GridLines → Automatic,
  Frame → True, FrameLabel → {"True θ, hours", "Accept Prob.",
    "Exact Operational-Characteristic Curve" , None},
  PlotStyle → RGBColor[0, 0, 1]];
```



Cumulative Acceptance Probabilities for Each Stage When the True θ Equals 1480 Hours

It would be useful to generate a list of cumulative acceptance probabilities for all six stages. The parameter *trueTheta* will be left symbolic in order to be consistent with up-front calculations. The desired list is generated but display of the output is temporarily suppressed.

```
mycumacc = Table[
  Sum[AcceptanceProbability[stage, trueTheta], {stage, 1, stagelim}], {stagelim, 1, 6}];
```

Now a table is generated which displays our calculations. A rule is used to replace *trueTheta* with 1480 hours in the cumulative acceptance probabilities stored in the list *mycumacc*:

```
NumberForm[
 TableForm[{N[timeValues, 4], N[mycumacc /. trueTheta → 1480, 4]},
  TableDirections -> {Row, Column},
  TableHeadings -> {{"Time", "Σ Accept Pr."}, Automatic},
  TableAlignments -> Center], {6, 5}]
```

	Time	Σ Accept Pr.
1.	2382.	0.2
2.	4432.	0.28056
3.	6333.	0.32939
4.	8162.	0.36386
5.	9947.	0.39021
6.	11701.	0.4114

We can observe that the probability of acceptance is 20% at the first stage and accumulates ultimately to approximately 41%.

In order to calculate just the final cumulative acceptance probability, we can use *OCfunction* from the previous section and employ a rule to replace *trueTheta* with 1480 hours.

```
OCfunction /. trueTheta → 1480
```

$$\frac{1020632231927244439}{17752052992000000\, e^{11701/1480}} + \frac{104896123050343}{4797852160000\, e^{9947/1480}} + \frac{2774933529}{324179200\, e^{4081/740}} + \frac{3860031}{1095200\, e^{6333/1480}} + \frac{1191}{740\, e^{554/185}} + \frac{1}{e^{1191/740}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.4114009662053622390212
```

This is the consumer risk since the consumer risk is defined as the acceptance probability when the true θ equals the lower-test θ.

- **Construct Function for Continuation Probability for a Quantity of Failures**

In this step, we construct a function for calculating continuation probabilities for a quantity of failures [Epstein, et al. 1963, equation 19]:

```
ContinuationProbability[stage_Integer, failure_Integer, trueTheta_] :=
 ACProbability[stage, failure, trueTheta]
```

- **Construct and Use Function for Continuation Probability for Each Stage**

In this step, we construct and use a function for calculating stage-by-stage continuation probabilities [Epstein, et al. 1963, equation 21]:

```
ContinuationProbability[stage_Integer, trueTheta_] :=
    r[stage]-1
     ∑          ContinuationProbability[stage, failure, trueTheta] /;
failure=a[stage]+1
  a[stage] + 1 < r[stage]

ContinuationProbability[stage_Integer, trueTheta_] :=
 0 /; Not[a[stage] + 1 < r[stage]]
```

The continuation probability for stage zero with zero failures is, by definition, one:

```
ContinuationProbability[0, trueTheta]
```

```
1
```

Cumulative Continuation Probabilities for Each Stage When the True θ is Symbolic

The cumulative continuation probability for stage one when *trueTheta* is left symbolic is:

```
ContinuationProbability[1, trueTheta]
```

$$\frac{3195196295609268\ e^{-2382/\text{trueTheta}}}{5\ \text{trueTheta}^5} + \frac{1341392231574\ e^{-2382/\text{trueTheta}}}{\text{trueTheta}^4} + \frac{2252547828\ e^{-2382/\text{trueTheta}}}{\text{trueTheta}^3} + \frac{2836962\ e^{-2382/\text{trueTheta}}}{\text{trueTheta}^2} + \frac{2382\ e^{-2382/\text{trueTheta}}}{\text{trueTheta}}$$

This is an exact symbolic result. An exact result for the case where *trueTheta* is 1480 hours is:

```
% /. trueTheta → 1480
```

$$\frac{35229506356658217}{8876026496000000\ e^{1191/740}}$$

This is an exact result. An numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.7937954509181623006374
```

The cumulative continuation probability for stage two when *trueTheta* is left symbolic is:

```
ContinuationProbability[2, trueTheta]
```

$$\frac{69741890180605268\ e^{-4432/\text{trueTheta}}}{5\ \text{trueTheta}^5} + \frac{15340486306474\ e^{-4432/\text{trueTheta}}}{\text{trueTheta}^4} + \frac{13073497428\ e^{-4432/\text{trueTheta}}}{\text{trueTheta}^3} + \frac{7720062\ e^{-4432/\text{trueTheta}}}{\text{trueTheta}^2}$$

This is an exact symbolic result. An exact result for the case where *trueTheta* is 1480 hours is:

```
% /. trueTheta → 1480
```

$$\frac{112894152208872217}{8876026496000000\ e^{554/185}}$$

This is an exact result. An numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.6366737562654589956167
```

The cumulative continuation probability for stage six when *trueTheta* is symbolic is:

```
ContinuationProbability[6, trueTheta]
```

```
0
```

This is clearly correct since the continuation probability at the last stage must be zero.

Cumulative Continuation Probabilities for Each Stage When the True θ Equals 1480 Hours

The cumulative continuation probability for stage one when *trueTheta* equals 1480 hours is:

```
ContinuationProbability[1, trueTheta] /. trueTheta → 1480
```

$$\frac{35229506356658217}{8876026496000000\ e^{1191/740}}$$

This is an exact result. An numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.7937954509181623006374
```

It would be useful to generate a list of cumulative continuation probabilities for all six stages. The parameter *trueTheta* will be left symbolic in order to be consistent with up-front calculations. The desired list is generated but display of the output is temporarily suppressed.

```
mycumcon =
  Table[ContinuationProbability[stage, trueTheta], {stage, 1, 6}];
```

Now a table is generated which displays our calculations. A rule is used to replace *trueTheta* with 1480 hours in the cumulative continuation probabilities stored in the list *mycumcon*. The cumulative acceptance probabilities for the case where the true θ equals 1480 hours are also provided for reference.

```
NumberForm[
 TableForm[{N[timeValues, 4], N[mycumacc /. trueTheta → 1480, 4],
   N[mycumcon /. trueTheta → 1480, 4]},
  TableDirections → {Row, Column}, TableHeadings ->
   {{"Time", "Σ Accept Pr.", "Σ Continue Pr."}, Automatic},
  TableAlignments -> Center], {6, 5}]
```

	Time	Σ Accept Pr.	Σ Continue Pr.
1.	2382.	0.2	0.7938
2.	4432.	0.28056	0.63667
3.	6333.	0.32939	0.42243
4.	8162.	0.36386	0.21336
5.	9947.	0.39021	0.0693
6.	11701.	0.4114	0.

- **Calculate Rejection Probability for Each Stage**

In this step, we calculate stage-by-stage rejection probabilities using Epstein, et al. 1963, equation 22:

$$\sum_{stage=1}^{n} (ContinuationProbability[stage - 1, trueTheta] - ContinuationProbability[stage, trueTheta] - AcceptanceProbability[stage, trueTheta])$$

Cumulative Rejection Probabilities for Each Stage When the True θ is Symbolic

The cumulative rejection probability for stage one when *trueTheta* is left symbolic is:

```
∑ (ContinuationProbability[stage - 1, trueTheta] -
stage=1
   ContinuationProbability[stage, trueTheta] -
   AcceptanceProbability[stage, trueTheta])
```

$$1 - e^{-2382/\text{trueTheta}} - \frac{3195196295609268\, e^{-2382/\text{trueTheta}}}{5\,\text{trueTheta}^5} - \frac{1341392231574\, e^{-2382/\text{trueTheta}}}{\text{trueTheta}^4} - \frac{2252547828\, e^{-2382/\text{trueTheta}}}{\text{trueTheta}^3} - \frac{2836962\, e^{-2382/\text{trueTheta}}}{\text{trueTheta}^2} - \frac{2382\, e^{-2382/\text{trueTheta}}}{\text{trueTheta}}$$

This is an exact, but partially symbolic result. An exact result can be obtained for a specific value of *trueTheta* such as 1480 hours is as follows:

```
% /. trueTheta → 1480
```

$$1 - \frac{44105532852658217}{8876026496000000\, e^{1191/740}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.0062088584404824196077 09
```

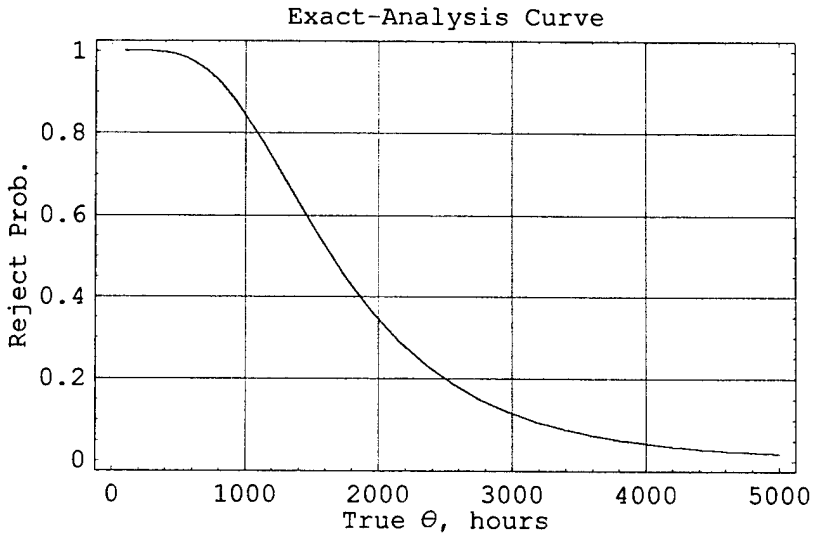
Next, the cumulative rejection probability for stage six when *trueTheta* is symbolic will be generated. This is one minus the operational-characteristic function.

```
rejfunction = Sum[stage=1 to 6] (ContinuationProbability[stage - 1, trueTheta] -
    ContinuationProbability[stage, trueTheta] -
    AcceptanceProbability[stage, trueTheta])
```

$$1 - e^{-2382/\text{trueTheta}} - \frac{204126446385448878\, e^{-11701/\text{trueTheta}}}{5\ \text{trueTheta}^5} - \frac{104896123050343\, e^{-9947/\text{trueTheta}}}{\text{trueTheta}^4} - \frac{27749335290\, e^{-8162/\text{trueTheta}}}{\text{trueTheta}^3} - \frac{7720062\, e^{-6333/\text{trueTheta}}}{\text{trueTheta}^2} - \frac{2382\, e^{-4432/\text{trueTheta}}}{\text{trueTheta}}$$

rejfunction provides the exact rejection probability as a function of *trueTheta*. This function can now be plotted:

```
Plot[rejfunction, {trueTheta, 100, 5000}, GridLines → Automatic,
  Frame → True, FrameLabel → {"True θ, hours", "Reject Prob.",
    "Exact-Analysis Curve" , None}, PlotStyle → RGBColor[0, 0, 1]];
```



Cumulative Rejection Probabilities for Each Stage When the True θ Equals 1480 Hours

It would be useful to generate a list of cumulative rejection probabilities for all six stages. The parameter *trueTheta* will be left symbolic in order to be consistent with up-front calculations. The desired list is generated but display of the output is temporarily suppressed.

```
mycumrej = Table[ Sum_{stage=1}^{stagelim} (ContinuationProbability[stage - 1, trueTheta] -
      ContinuationProbability[stage, trueTheta] -
      AcceptanceProbability[stage, trueTheta]), {stagelim, 1, 6}];
```

Now a table is generated which displays our calculations. A rule is used to replace *trueTheta* with 1480 hours in the cumulative rejection probabilities stored in the list *mycumrej*. The cumulative acceptance and continuation probabilities for the case where the true θ equals 1480 hours are also provided for reference.

```
NumberForm[
 TableForm[{N[timeValues, 4], N[mycumacc /. trueTheta → 1480, 4], N[
    mycumcon /. trueTheta → 1480, 4], N[mycumrej /. trueTheta → 1480, 4]},
  TableDirections -> {Row, Column}, TableHeadings ->
   {{"Time", "Σ Accept Pr.", "Σ Continue Pr.", "Σ Reject Pr."},
    Automatic}, TableAlignments -> Center], {6, 5}]
```

	Time	Σ Accept Pr.	Σ Continue Pr.	Σ Reject Pr.
1.	2382.	0.2	0.7938	0.00621
2.	4432.	0.28056	0.63667	0.08277
3.	6333.	0.32939	0.42243	0.24818
4.	8162.	0.36386	0.21336	0.42278
5.	9947.	0.39021	0.0693	0.54048
6.	11701.	0.4114	0.	0.5886

Each row in the table above sums to one as it should. In order to calculate just the final cumulative rejection probability, we can use the *rejfunction* from the previous section and employ a rule to replace *trueTheta* with 1480 hours.

```
rejfunction /. trueTheta → 1480
```

$$1 - \frac{1020632231927244439}{17752052992000000\, e^{11701/1480}} - \frac{104896123050343}{4797852160000\, e^{9947/1480}} - \frac{2774933529}{324179200\, e^{4081/740}} - \frac{3860031}{1095200\, e^{6333/1480}} - \frac{1191}{740\, e^{554/185}} - \frac{1}{e^{1191/740}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.5885990337946377609788
```

The consumer risk equals one minus the rejection probability when the true θ equals the required θ. This is one minus the answer above:

```
1 - %
```

```
0.411400966205362239021 2
```

It would also be helpful to overlay the accept and reject probabilities on the decision-rule plot generated earlier for the case where the true θ equals 1480 hours. First a graphics object will be generated but temporarily suppressed for the accept points. The accept points will be represented by triangles.

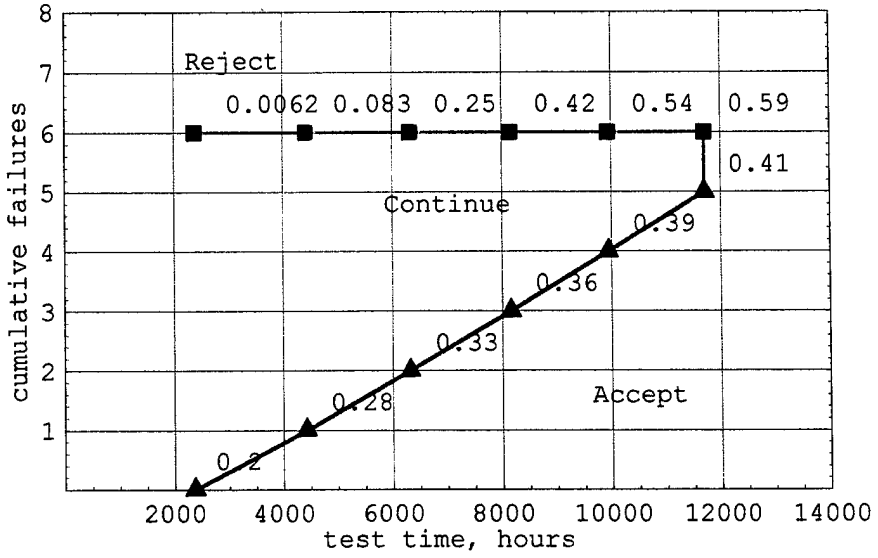
```
acceptProbPtsPlot = MultipleListPlot[Table[{N[t[stage]], a[stage]},
    {stage, 1, Length[timeValues]}], SymbolShape ->
  PlotSymbol[Triangle, 5], SymbolStyle → RGBColor[0, 0, 0], SymbolLabel →
    {Map[NumberForm[#, 2] &, N[mycumacc /. trueTheta → 1480]], None},
   PlotRange → {{0, 14000}, {0, 6.8}}, Frame → True,
   FrameLabel → {"test time, hours", "cumulative failures"},
   GridLines → Automatic, DisplayFunction → Identity];
```

Next a graphics object will be generated but temporarily suppressed for the reject points. The reject points will be represented by boxes.

```
rejectProbPtsPlot = MultipleListPlot[Table[{N[t[stage]], r[stage]},
    {stage, 1, Length[timeValues]}], SymbolShape ->
  PlotSymbol[Box, 3], SymbolStyle → RGBColor[0, 0, 0], SymbolLabel →
    {Map[NumberForm[#, 2] &, N[mycumrej /. trueTheta → 1480]], None},
   PlotRange → {{0, 14000}, {0, 6.8}}, Frame → True,
   FrameLabel → {"test time, hours", "cumulative failures"},
   GridLines → Automatic, DisplayFunction → Identity];
```

Now we will display the accept and reject points overlaid on the decision-rule plot.

```
Show[decisionPlot, acceptProbPtsPlot, rejectProbPtsPlot,
  PlotRange → {{0, 14000}, {0, 8}}, DisplayFunction → $DisplayFunction];
```



- **Calculate Expected Quantity of Failures and Test Time**

We need to define a function for the probability that the test will terminate with an accept decision at a specified number of failures [Epstein, et al. 1963, equation 33]. First the general case and then the special case:

```
AccProbabilityF[failure_Integer, trueTheta_] :=
 Module[{stage = 1}, While[failure > a[stage], stage++]; Which[
    failure > Last[Last[accept]], 0, 0 ≤ failure ≤ Last[Last[accept]],
    AcceptanceProbability[stage, failure, trueTheta]]] /;
  failure ≤ a[Length[timeValues]]

AccProbabilityF[failure_Integer, trueTheta_] :=
 0 /; failure > a[Length[timeValues]]
```

Now, we will define a function for the probability that the test will terminate with a reject decision at a specified number of failures [Epstein, et al. 1963 equation 34]:

```
RejProbabilityF[failure_Integer, trueTheta_] :=
 Module[{rejectlist}, rejectlist =
   Select[Table[{stage, r[stage]}, {stage, 1, Length[timeValues]}],
     #[[2]] == failure &] /. {st_Integer, rej_Integer} → st;
  Which[Length[rejectlist] == 0, 0, Length[rejectlist] > 0,
   Sum[(ContinuationProbability[stage - 1, trueTheta] -
      ContinuationProbability[stage, trueTheta] -
      AcceptanceProbability[stage, trueTheta]),
    {stage, First[rejectlist], Last[rejectlist]}]]]
```

The probability that the test will terminate with zero failures and a reject decision is:

```
RejProbabilityF[0, trueTheta]
```

0

This is obviously correct since the only path to rejection is if six failures occurs. The probability that the test will terminate with one failure through five failures and a reject decision is:

```
RejProbabilityF[1, trueTheta]
```

0

```
RejProbabilityF[2, trueTheta]
```

0

```
RejProbabilityF[3, trueTheta]
```

0

```
RejProbabilityF[4, trueTheta]
```

0

```
RejProbabilityF[5, trueTheta]
```

0

The probability that the test will terminate with six failures and a reject decision is:

```
RejProbabilityF[6, trueTheta]
```

$$1 - e^{-2382/\text{trueTheta}} - \frac{2041264463854488878\, e^{-11701/\text{trueTheta}}}{5\,\text{trueTheta}^5} - \frac{104896123050343\, e^{-9947/\text{trueTheta}}}{\text{trueTheta}^4} - \frac{27749335290\, e^{-8162/\text{trueTheta}}}{\text{trueTheta}^3} - \frac{7720062\, e^{-6333/\text{trueTheta}}}{\text{trueTheta}^2} - \frac{2382\, e^{-4432/\text{trueTheta}}}{\text{trueTheta}}$$

If *trueTheta* is equal to 1480 hours, the probability that the test will terminate with six failures and a reject decision is:

```
% /. trueTheta → 1480
```

$$1 - \frac{1020632231927244439}{17752052992000000\, e^{11701/1480}} - \frac{104896123050343}{4797852160000\, e^{9947/1480}} - \frac{2774933529}{324179200\, e^{4081/740}} - \frac{3860031}{1095200\, e^{6333/1480}} - \frac{1191}{740\, e^{554/185}} - \frac{1}{e^{1191/740}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.5885990337946377609788
```

Now, we will define a function for the probability that the test will terminate in either acceptance or rejection with a specified number of failures [Epstein, et al. 1963 equation 35]:

```
TerminateProbability[failure_Integer, trueTheta_] :=
 AccProbabilityF[failure, trueTheta] +
  RejProbabilityF[failure, trueTheta]
```

The probability that the test will terminate with zero failures is:

```
TerminateProbability[0, trueTheta]
```

$$e^{-2382/\text{trueTheta}}$$

The probability that the test will terminate with one failure is:

```
TerminateProbability[1, trueTheta]
```

$$\frac{2382\, e^{-4432/\text{trueTheta}}}{\text{trueTheta}}$$

The probability that the test will terminate with between zero and six failures is:

$$\sum_{failure=0}^{6} \texttt{TerminateProbability[failure, trueTheta]}$$

1

This result is obviously correct since it's not possible for the test to continue beyond the sixth failure.

Next, we will define a function for the expected termination failure quantity [Epstein, et al. 1963 equation 36]:

$$\texttt{ExpectedTerminationFailure[trueTheta\_] :=}$$
$$\sum_{failure=0}^{r[Length[timeValues]]} \texttt{failure TerminateProbability[failure, trueTheta]}$$

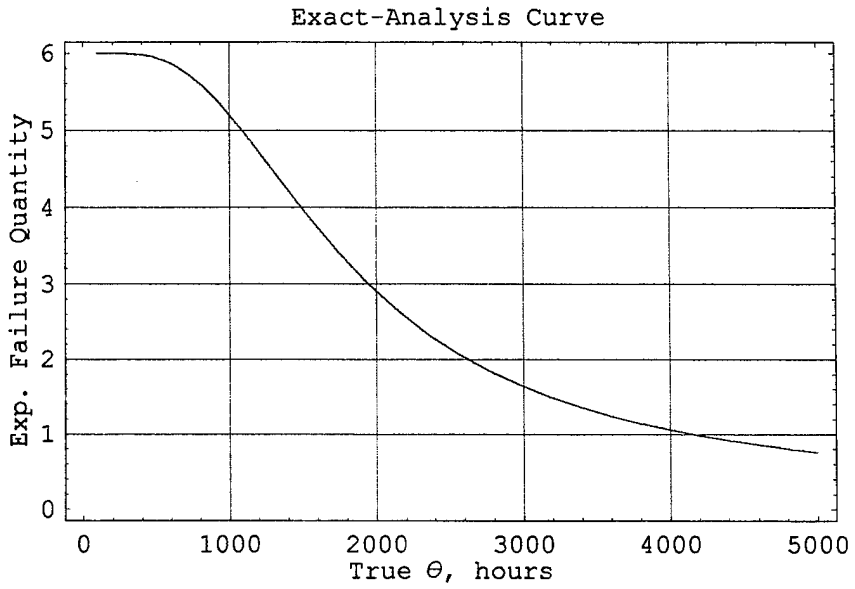
A function for the expected termination failure quantity with *trueTheta* left symbolic is:

$$\texttt{expectedfailurefunction = ExpectedTerminationFailure[trueTheta]}$$

$$6\left(1 - e^{-2382/\text{trueTheta}} - \frac{2041264463854488878\, e^{-11701/\text{trueTheta}}}{5\,\text{trueTheta}^5} - \frac{104896123050343\, e^{-9947/\text{trueTheta}}}{\text{trueTheta}^4} - \frac{27749335290\, e^{-8162/\text{trueTheta}}}{\text{trueTheta}^3} - \frac{7720062\, e^{-6333/\text{trueTheta}}}{\text{trueTheta}^2} - \frac{2382\, e^{-4432/\text{trueTheta}}}{\text{trueTheta}}\right) + \frac{2041264463854488878\, e^{-11701/\text{trueTheta}}}{\text{trueTheta}^5} + \frac{419584492201372\, e^{-9947/\text{trueTheta}}}{\text{trueTheta}^4} + \frac{83248005870\, e^{-8162/\text{trueTheta}}}{\text{trueTheta}^3} + \frac{15440124\, e^{-6333/\text{trueTheta}}}{\text{trueTheta}^2} + \frac{2382\, e^{-4432/\text{trueTheta}}}{\text{trueTheta}}$$

Now we can plot this function:

```
Plot[expectedfailurefunction,
  {trueTheta, 100, 5000}, GridLines → Automatic, Frame → True,
  FrameLabel → {"True θ, hours", "Exp. Failure Quantity",
    "Exact-Analysis Curve" , None}, PlotStyle → RGBColor[0, 0, 1]];
```



In order to calculate the expected failure quantity for a true θ of 1480 hours, we could use *expectedfailurefunction* and a rule to replace *trueTheta* with 1480.

```
expectedfailurefunction /. trueTheta → 1480
```

$$6\left(1-\frac{1020632231927244439}{17752052992000000\,e^{11701/1480}}-\frac{104896123050343}{4797852160000\,e^{9947/1480}}-\frac{2774933529}{324179200\,e^{4081/740}}-\frac{3860031}{1095200\,e^{6333/1480}}-\frac{1191}{740\,e^{554/185}}-\frac{1}{e^{1191/740}}\right)+\frac{1020632231927244439}{3550410598400000\,e^{11701/1480}}+\frac{104896123050343}{1199463040000\,e^{9947/1480}}+\frac{8324800587}{324179200\,e^{4081/740}}+\frac{3860031}{547600\,e^{6333/1480}}+\frac{1191}{740\,e^{554/185}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
4.024572455177141012839
```

Next, we will define a function for the expected test time [Epstein, et al. 1963 equation 41]:

```
ExpectedTestTime[trueTheta_] :=
 trueTheta ExpectedTerminationFailure[trueTheta]
```

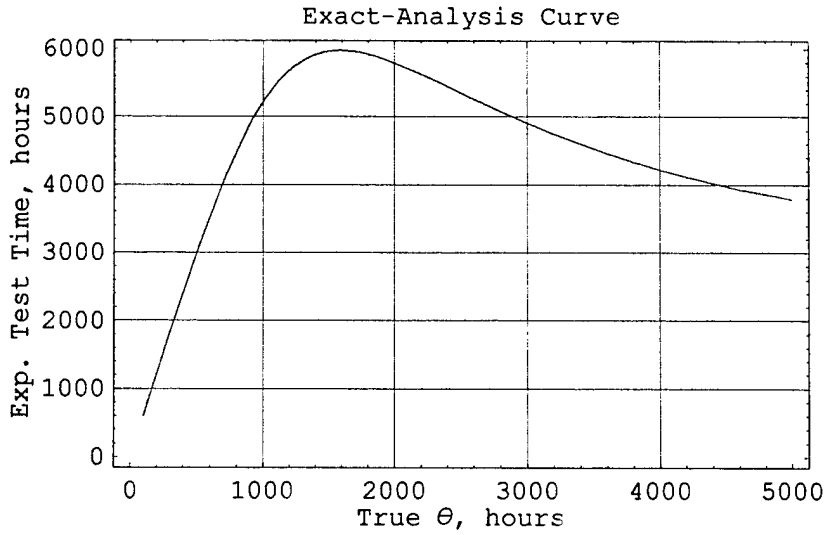
A function for the expected test time with *trueTheta* left symbolic is:

```
expectedtesttimefunction = ExpectedTestTime[trueTheta]
```

$$\left(6\left(1-e^{-2382/\text{trueTheta}}-\frac{2041264463854488878\,e^{-11701/\text{trueTheta}}}{5\,\text{trueTheta}^5}-\frac{104896123050343\,e^{-9947/\text{trueTheta}}}{\text{trueTheta}^4}-\frac{27749335290\,e^{-8162/\text{trueTheta}}}{\text{trueTheta}^3}-\frac{7720062\,e^{-6333/\text{trueTheta}}}{\text{trueTheta}^2}-\frac{2382\,e^{-4432/\text{trueTheta}}}{\text{trueTheta}}\right)+\frac{2041264463854488878\,e^{-11701/\text{trueTheta}}}{\text{trueTheta}^5}+\frac{419584492201372\,e^{-9947/\text{trueTheta}}}{\text{trueTheta}^4}+\frac{83248005870\,e^{-8162/\text{trueTheta}}}{\text{trueTheta}^3}+\frac{15440124\,e^{-6333/\text{trueTheta}}}{\text{trueTheta}^2}+\frac{2382\,e^{-4432/\text{trueTheta}}}{\text{trueTheta}}\right)\text{trueTheta}$$

Now we can plot this function:

```
Plot[expectedtesttimefunction,
  {trueTheta, 100, 5000}, GridLines → Automatic, Frame → True,
  FrameLabel → {"True θ, hours", "Exp. Test Time, hours",
    "Exact-Analysis Curve" , None}, PlotStyle → RGBColor[0, 0, 1]];
```



In order to calculate the expected test time for a true θ of 1480 hours, we could use *expectedtesttimefunction* and a rule to replace *trueTheta* with 1480.

```
expectedtesttimefunction /. trueTheta → 1480
```

$$1480\left(6\left(1-\frac{1020632231927244439}{17752052992000000\,e^{11701/1480}}-\frac{104896123050343}{4797852160000\,e^{9947/1480}}-\frac{2774933529}{324179200\,e^{4081/740}}-\frac{3860031}{1095200\,e^{6333/1480}}-\frac{1191}{740\,e^{554/185}}-\frac{1}{e^{1191/740}}\right)+\frac{1020632231927244439}{3550410598400000\,e^{11701/1480}}+\frac{104896123050343}{1199463040000\,e^{9947/1480}}+\frac{8324800587}{324179200\,e^{4081/740}}+\frac{3860031}{547600\,e^{6333/1480}}+\frac{1191}{740\,e^{554/185}}\right)$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
5956.367233662168699002
```

Summary & Conclusions

An analysis of the exact stage-by-stage acceptance, continuation and rejection probabilities, including the impact of truncation, resulting from a sequence of reliability test plan decision rules was performed. An exact operational-characteristic function was obtained and plotted. Exact functions for expected failure quantity and test time were obtained and plotted as well.

The probability of accepting a product with a true θ or MTBF equal to the lower-test value of 1480 hours at the first stage is 20%. If acceptance does not occur at the first stage, there is a possibility of acceptance at each subsequent stage except the last. The total probability of accepting such a product (i.e., the consumer risk) is the sum of the acceptance probabilities at each stage. The consumer risk was found to be 41.1%, approximately twice the desired 20% value.

The sequence of decision rules analyzed herein was also simulated for 400,000 trials in Appendix B. The results of the simulated tests are consistent with the results obtained in this notebook and thus constitute a rough double-check of the exact analysis.

The interpretation of the procedure from [Kececioglu 1993, section 7.10] considered herein is thus rejected. As a result, a contractor-proposed test plan that contained this serious error was rejected. This chapter illustrates this mistake and its impact in the hope this may help test designers avoid it in the future.

It should also be noted that Butler and Lieberman (1980) recognized that a plan developed as described in the beginning of this chapter would have a total consumer risk roughly twice the consumer risk at the first accept point, as we saw. They developed a method for designing a sequential test plan containing multiple acceptance points and a single, final reject point that has a more desirable level of consumer risk.

Chapter 4

Design & Analysis of a Truncated Exponential Sequential Test: Case Study 3

Introduction

Test planners were considering various options for a reliability demonstration test on a tactical terminal. It was requested that AMSAA develop a truncated exponential sequential test plan that would meet the following requirements:

- lower-test MTBF = 2200 hours
- upper-test MTBF = 4400 hours
- consumer-risk goal = producer-risk goal = 20%

Sequential test designs can often reduce test time by a factor of two compared with fixed-length designs thus they are highly beneficial to use when practical.

This chapter documents the design of a truncated exponential sequential test design that approximately meets the requirements above. The impact of the truncation can only be determined through subsequent analysis of the test design. For this reason, an exact analysis was performed and may be found in Appendix C. Key results of the exact analysis, including stage-by-stage acceptance, continuation and rejection probabilities, actual consumer and producer risks and operational-characteristic curve, are included in this chapter. The exact analysis was checked with a simulation which is not included in this report due to considerations of report length.

Define Requirements

• Lower-test MTBF = 2200 hours. We'll assign this as the value of the symbol *lowertest* since it will be used numerous times in this example.

```
lowertest = 2200
```

```
2200
```

• Upper-test MTBF = 4400 hours. We'll assign this as the value of the symbol *uppertest*.

```
uppertest = 4400
```

```
4400
```

• Consumer risk = 20%. We'll assign this as the value of the symbol *conrisk*. First we'll convert 20% to an exact rational number. If we specify any input as a Real number, *Mathematica* will use finite-precision arithmetic for each calculation. Errors will propagate and grow with each successive calculation. In order to avoid inaccuracies in statistical analyses, McCullough [2000] recommends that Real inputs be converted to Rationals. If we convert Reals to Rationals, exact arithmetic will be used (i.e., no approximations or roundoffs will occur). *Mathematica* will provide approximate numerical results only when the user insists.

```
conrisk = Rationalize[.2]
```

$$\frac{1}{5}$$

• Producer risk = 20%. We'll assign this as the value of the symbol *prodrisk*.

```
prodrisk = Rationalize[.2]
```

$$\frac{1}{5}$$

Setup

The code to be used in this chapter is contained in the new package *ExponentialSequentialTestDesign.m* (Appendix D) and the standard add-on packages *MultipleListPlot.m* and *DiscreteDistributions.m*, These packages are loaded thus:

```
Needs["Statistics`DiscreteDistributions`"]
```

```
Needs["Graphics`MultipleListPlot`"]
```

```
Needs["Reliability`ExponentialSequentialTestDesign`"]
```

The version of *ExponentialSequentialTestDesign.m* used in this test design is determined next.

```
? ExponentialSequentialTestDesign

ExponentialSequentialTestDesign.m (version 0.8.0) is a
  package which contains a collection of functions useful for
  sequential test design based on the Exponential distribution.
```

Accept Function

First, we need an accept function. The accept function provides cumulative failures for acceptance as a function of cumulative test time, given values for lower-test MTBF, consumer risk, upper-test MTBF and producer risk. The new function `ExponentialAccept` implements the accept equation derived by Epstein and Sobel (1955). Providing the values in this case as arguments to `ExponentialAccept` and leaving the argument *time* symbolic yields:

```
acceptfun =
 ExponentialAccept[lowertest, conrisk, uppertest, prodrisk, time]
```

$$\frac{\text{time}}{4400\,\text{Log}[2]} - \frac{\text{Log}[4]}{\text{Log}[2]}$$

This is the function for the non-truncated segment of the accept line. One use of the accept function is to calculate the minimum test length for an accept decision to be reached (i.e., the quantity of time that, if reached before a single failure occurs, triggers an accept decision). Acceptance occurs between failures if the accumulated test time meets or exceeds the corresponding value given by the accept function. The shortest path to acceptance can be determined by setting the quantity of failures equal to zero and solving for *time*.

```
Solve[acceptfun == 0, time] // N

{{time → 6099.7}}
```

Thus, the shortest path to an accept decision is when no failures have occurred and the accumulated test time ≥ 6099.7 hours.

Reject Function

Next, we need a function for the non-truncated segment of the reject line. The reject function provides cumulative failures for rejection as a function of cumulative test time, given values for lower-test MTBF, consumer risk, upper-test MTBF and producer risk. The Wald intercept, an approximation, will be used for this line.The new function `ExponentialReject` implements the reject equation derived by Epstein and Sobel (1955). Providing the values of the arguments in this example to `ExponentialReject`, leaving the argument *time* symbolic and setting the option `ConstantAMethod →Wald` yields:

```
rejectfun = ExponentialReject[lowertest, conrisk,
  uppertest, prodrisk, time, ConstantAMethod → Wald]
```

$$\frac{\text{time}}{4400\,\text{Log}[2]} + \frac{\text{Log}[4]}{\text{Log}[2]}$$

This is the function for the non-truncated segment of the reject line. One use of the reject function is to calculate the minimum test length for a reject decision to be reached. A rejection decision is triggered by the occurrence of a failure. When a failure occurs, if the accumulated test time is less than or equal to the corresponding value given by the reject function, rejection occurs. The shortest path to failure is the smallest quantity of failures which results in a positive rejection time. This can be determined by setting the quantity of failures equal to zero and solving for *time*.

```
Solve[rejectfun == 0, time] // N
```

```
{{time → -6099.7}}
```

The quantity of time corresponding to zero failures is negative which is not a physically meaningful answer. We will increment the failure quantity and repeat this calculation until we first obtain a positive quantity of time:

```
Solve[rejectfun == 3, time] // N
```

```
{{time → 3049.85}}
```

The shortest path to rejection occurs if the third failure occurs and the accumulated test time ≤ 3049.85 hours.

Truncation

Next, we'll address the truncation rules. First we'll calculate the truncated-reject criterion using the Epstein [1954] truncation method which is implemented by the new function `ExponentialTruncationFailures`. We'll assign the result as the value of the symbol *failtrunc*:

```
failtrunc =
 ExponentialTruncationFailures[lowertest, conrisk, uppertest, prodrisk]
```

```
7
```

Now, we'll calculate the corresponding Epstein truncated-accept criterion *timetrunc* by using the new function `ExponentialTruncationTime` with the upper-test MTBF = *uppertest*, producer risk = *prodrisk* and *failures* = *failtrunc*. We'll do this so as to avoid excess test time. In general, when the accept time for the (*failtrunc* - 1) failure is less than the value of *timetrunc*, excess test time exists (i.e., test time which it would be impossible to reach). This quantity of excess test time is misleading and it is desirable to remove it. In order to avoid erroneous test time, we first calculate the accept time for the (*failtrunc* - 1) failure and assign the result as the value of *oneminusacceptime*:

```
N[oneminusaccepttime =
  First[time /. Solve[acceptfun == failtrunc - 1, time]]]
```

```
24398.8
```

Next we calculate the Epstein truncation time and assign the result as the value of *epsteintimetrunc*:

```
N[epsteintimetrunc =
  ExponentialTruncationTime[uppertest, prodrisk, failtrunc]]
```

```
20828.1
```

If *oneminusacceptime* is less than *epsteintimetrunc*, we will assign *oneminusacceptime* as the value of *timetrunc*. Otherwise, we will assign *epsteintimetrunc* as the value of *timetrunc*:

```
If[oneminusaccepttime < epsteintimetrunc,
  timetrunc = oneminusaccepttime, timetrunc = epsteintimetrunc] // N
```

```
20828.1
```

Determine Domain of Functions and Generate Decision-Rule Plot

Now we are almost ready to plot the test plan. We will generate, but temporarily suppress, a graphics object for each of the four straight lines needed on the final plot (i.e., non-truncated accept, non-truncated reject, truncated accept and truncated reject). Then we will plot the graphics objects together. It may be helpful to refer to the plot at the end of this example as we proceed.

First, let's generate a graphics object for the non-truncated segment of the accept line using the `ExponentialAccept` function. We will limit the independent variable *time* to the domain of zero and *timetrunc* (where the truncated segment of the accept line begins) and we will use the `PlotRange` option to restrict the range of the function so that it will not plot below zero:

```
acceptPlot = Plot[acceptfun, {time, 0, timetrunc},
   PlotStyle → RGBColor[0, 1, 0], DisplayFunction → Identity];
```

Next, let's generate a graphics object for the non-truncated segment of the reject line using the `ExponentialReject` function with the Wald intercept selected. First, we need to determine where to truncate the reject line along the time axis (i.e., where the truncated segment of the reject line begins). We can determine this by using the `ExponentialReject` function, leaving the *time* argument symbolic, and solving for the quantity of time which corresponds to the truncation value of *failtrunc* failures.

```
N[rejecttimetrunc = First[time /. Solve[rejectfun == failtrunc, time]]]
```

```
15249.2
```

Now we can generate the graphics object for the non-truncated segment of the reject line using the `ExponentialReject` function while limiting the independent variable *time* to the domain of zero and *rejecttimetrunc* hours. We will use the `PlotRange` option to trigger plotting of the range of the function so that it is consistent with the generation of the other graphics objects.

```
rejectWaldPlot = Plot[rejectfun, {time, 0, rejecttimetrunc},
   PlotStyle → RGBColor[1, 0, 0], DisplayFunction → Identity];
```

Next, let's generate a graphics object for the truncated segment of the reject line. Since we reject at *failtrunc* failures between *rejecttimetrunc* and *timetrunc* hours, the truncated segment of the reject line is a horizontal line at *failtrunc* failures over this domain:

```
rejectTruncationPlot =
  Plot[failtrunc, {time, rejecttimetrunc, timetrunc},
   PlotStyle → RGBColor[1, 0, 0], DisplayFunction → Identity];
```

Now we need to generate a graphics object for the truncated segment of the accept line. This segment is a vertical line at *timetrunc* hours which extends from the end of the non-truncated segment of the accept line to the truncated-reject criterion of *failtrunc* failures. Before we can generate this graphics object, we must determine where the non-truncated segment of the accept line ends in terms of the failure axis. We can obtain this failure quantity by using `ExponentialAccept` and supplying *timetrunc* hours as the *time* argument.

```
N[acceptfailtrunc = acceptfun /. time → timetrunc]
```

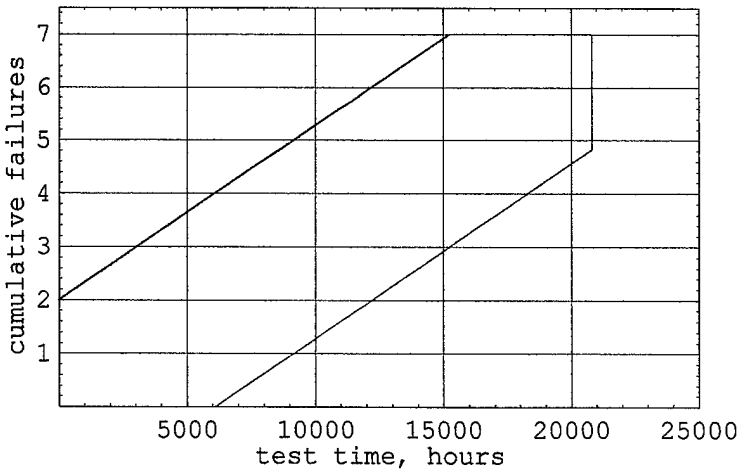
```
4.82923
```

Now we can generate a graphics object for the truncated segment of the accept line. This is a vertical line at *timetrunc* hours between *acceptfailtrunc* and *failtrunc* failures.

```
acceptTruncationPlot = Graphics[{RGBColor[0, 1, 0],
    Line[{{timetrunc, acceptfailtrunc}, {timetrunc, failtrunc}}]}];
```

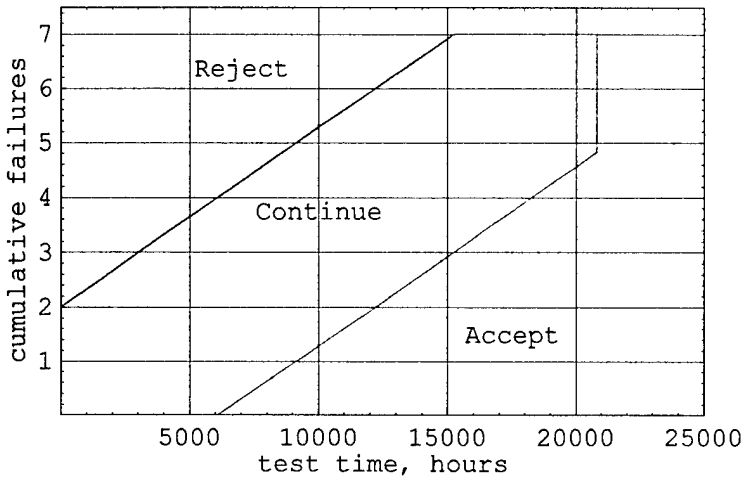
The final plot is obtained by displaying all of the graphics objects together:

```
Show[acceptPlot, rejectWaldPlot, rejectTruncationPlot,
  acceptTruncationPlot, PlotRange → {{0, 25000}, {0, 7.5}}, Frame → True,
  FrameLabel → {"test time, hours", "cumulative failures"},
  GridLines → Automatic, DisplayFunction → $DisplayFunction];
```



Identification of the reject, continue and accept regions can be overlaid thus:

```
decisionPlot = Show[%,
   Graphics[{Text["Reject", Scaled[{0.28, 0.85}]], Text["Continue",
      Scaled[{0.4, 0.5}]], Text["Accept", Scaled[{0.7, 0.2}]]}]];
```



Tabulate Accept and Reject Times

It would also be helpful to generate a table of decision rules for this test plan. First, we will generate a list of failure quantities but temporarily suppress the display since it will be presented in a table below (where it will be the first column):

```
failurelist = Range[0, failtrunc];
```

Next, we will generate a list of accept points calculated as discussed above for the accept-time column.

```
accepteqn = First[time /. Solve[acceptfun == r, time]];
```

```
acceptlist = Flatten[{Table[Which[(accepteqn /. r → failure) ≤ timetrunc,
      accepteqn /. r → failure, (accepteqn /. r → failure) > timetrunc,
      timetrunc], {failure, 0, failtrunc - 1}], "NA"}];
```

Next, we will generate a list of reject points calculated as discussed above for the reject-time column.

```
rejecteqn = First[time /. Solve[rejectfun == r, time]];
```

```
rejectlist = Flatten[{Table[Which[N[(rejecteqn /. r → failure)] ≤ 0, "NA",
      (rejecteqn /. r → failure) ≤ timetrunc, (rejecteqn /. r → failure),
      (rejecteqn /. r → failure) > timetrunc, timetrunc],
     {failure, 0, failtrunc - 1}], timetrunc}];
```

Now we display the entire table:

```
TableForm[Transpose[{failurelist, N[rejectlist], N[acceptlist]}],
 TableHeadings → {None, {"Failures", "Reject Time (hours) ≤",
    "Accept Time, (hours) ≥"}}, TableAlignments → Center]
```

Failures	Reject Time (hours) ≤	Accept Time, (hours) ≥
0	NA	6099.7
1	NA	9149.54
2	NA	12199.4
3	3049.85	15249.2
4	6099.7	18299.1
5	9149.54	20828.1
6	12199.4	20828.1
7	20828.1	NA

Rejection is triggered by the occurrence of a failure provided the accumulated test time is less than or equal to the value specified in the second column above. Acceptance occurs between failures if the accumulated test time meets or exceeds the values specified in the third column above.

Overlay Accept and Reject Times on Decision-Rule Plot

It would be helpful to overlay the decision points from the table above on the decision-rule plot. First a graphics object will be generated but temporarily suppressed for the reject points (excluding any points which are not numerical).

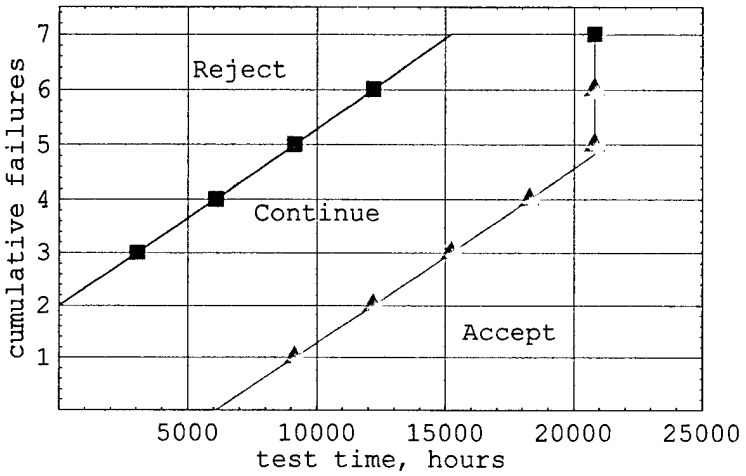
```
rejectPtsPlot = MultipleListPlot[
   Select[Transpose[{rejectlist, failurelist}], (#[[1]] > 0 && #[[2]] > 0) &],
   SymbolShape → PlotSymbol[Box, 3],
   SymbolStyle → RGBColor[1, 0, 0], DisplayFunction → Identity];
```

Next a graphics object will be generated but temporarily suppressed for the accept points (excluding any points which are not numerical).

```
acceptPtsPlot = MultipleListPlot[
   Select[Transpose[{acceptlist, failurelist}], (#[[1]] > 0 && #[[2]] > 0) &],
   SymbolShape → PlotSymbol[Triangle, 5],
   SymbolStyle → RGBColor[0, 1, 0], DisplayFunction → Identity];
```

Now we will display the accept and reject points, represented by triangles and boxes, respectively, overlaid on the decision-rule plot.

```
Show[decisionPlot, rejectPtsPlot,
  acceptPtsPlot, PlotRange → {{0, 25000}, {0, 7.5}}];
```

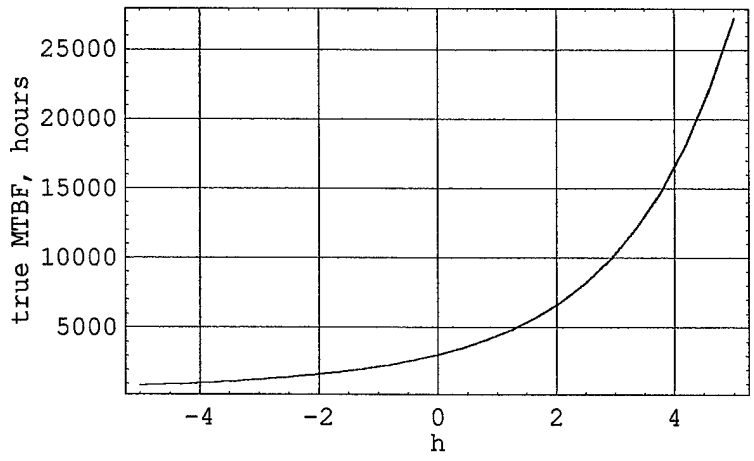


Approximate Operational-Characteristic Curve

It is also customary to plot the operational-characteristic function (i.e., acceptance probability as a function of true MTBF) for a test design. Epstein and Sobel (1955) derived a pair of equations which can be used to obtain an approximate operational-characteristic function. This function is approximate even in the untruncated case and does not account for truncation at all. The more truncated the test plan, the less accurate the approximation.

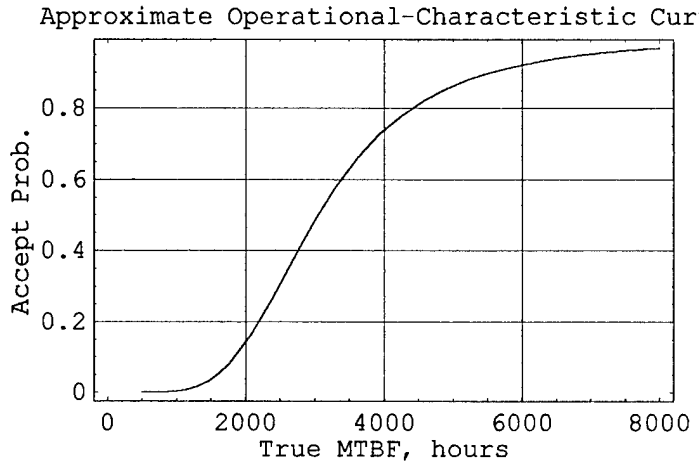
Two new add-on functions are based on these approximate Epstein and Sobel equations, `ExponentialAcceptProbability` and `ExponentialTrueMTBF`. Obtaining answers from `ExponentialAcceptProbability` as a function of the true MTBF requires that values for the exponent h be obtained from `ExponentialTrueMTBF`. Numerical root-finding is used to obtain values for h since `ExponentialTrueMTBF` is an implicit equation in terms of h (i.e., h cannot be isolated on the left-hand side of the equation and solved for analytically). `ExponentialTrueMTBF` will do this automatically provided a reasonable starting point is provided for the numerical root-finding algorithm. An easy way to obtain a reasonable starting point is by plotting `ExponentialTrueMTBF` versus h as follows:

```
Plot[ExponentialTrueMTBF[lowertest, uppertest, h], {h, -5, 5},
 PlotRange → All, Frame → True, FrameLabel → {"h", "true MTBF, hours"},
 PlotStyle → RGBColor[0, 0, 1], GridLines → Automatic];
```



Upon inspection of the plot above, it would seem that -4 would be a reasonable starting point for true MTBF values in the 1000 to 8000 hours range. It is important to avoid zero as a starting point since the equation is indeterminate there. Now we can generate an approximate operational-characteristic function by plotting `ExponentialAcceptProbability` versus true MTBF using the starting point just obtained.

```
approxOC = Plot[ExponentialAcceptProbability[
    lowertest, conrisk, uppertest, prodrisk, trueMTBF, -4],
  {trueMTBF, 500, 8000}, PlotRange → Automatic, Frame → True,
  FrameLabel → {"True MTBF, hours", "Accept Prob.",
    "Approximate Operational-Characteristic Curve" , None},
  PlotStyle → RGBColor[0, 0, 1], GridLines → Automatic];
```



Key Results from Exact Analysis

An exact analysis was performed and may be found in Appendix C. Key results are included in this section for examination of the test design. The stage-by-stage acceptance, continuation and rejection probabilities assuming the true MTBF equals the lower-test MTBF are:

ltMTBFtable

	Time	Σ Accept Pr.	Σ Continue Pr.	Σ Reject Pr.
1.	3050.	0.	0.83678	0.16322
2.	6100.	0.06249	0.60776	0.32975
3.	9150.	0.10581	0.43781	0.45638
4.	12199.	0.13584	0.31504	0.54912
5.	15249.	0.15709	0.22658	0.61632
6.	18299.	0.17227	0.10829	0.71944
7.	20828.	0.22243	0.	0.77757

Each row in the table above sums to one as it should. The acceptance probability at the last stage (i.e., the consumer risk) is approximately 22.2% which is fairly close to the consumer-risk goal of 20%. It's not possible to get exactly the desired consumer or producer risk when truncating.

The stage-by-stage acceptance, continuation and rejection probabilities assuming the true MTBF equals the upper-test MTBF are:

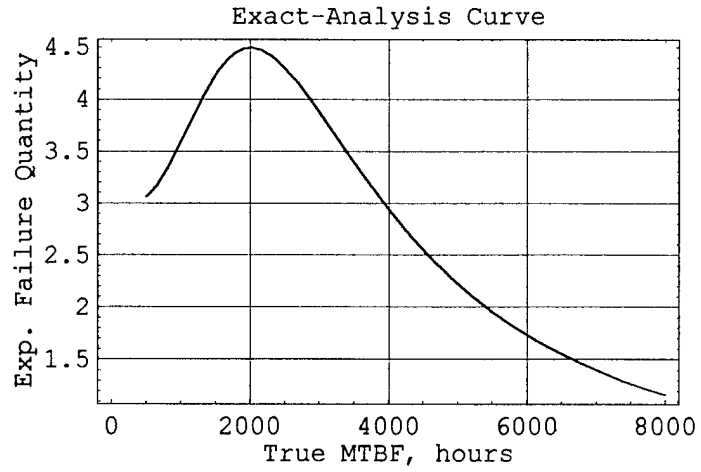
utMTBFtable

	Time	Σ Accept Pr.	Σ Continue Pr.	Σ Reject Pr.
1.	3050.	0.	0.96668	0.03332
2.	6100.	0.24998	0.68394	0.06608
3.	9150.	0.42326	0.48583	0.09091
4.	12199.	0.5434	0.34752	0.10909
5.	15249.	0.62839	0.24936	0.12225
6.	18299.	0.6891	0.15182	0.15907
7.	20828.	0.80273	0.	0.19727

The rejection probability at the last stage (i.e., the producer risk) is approximately 19.7%. This is quite close to the producer-risk goal of 20%.

The expected quantity of failures as a function of true MTBF is:

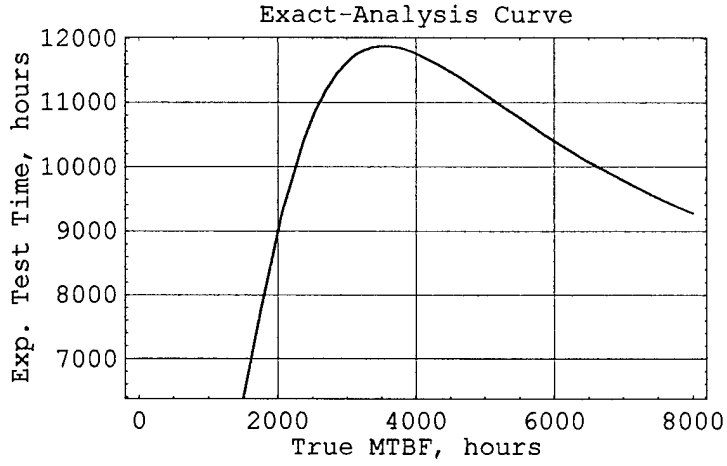
```
Show[expectedfailuresPlot];
```



The expected quantity of failures if the true MTBF equals the lower-test MTBF is 4.46. The expected quantity of failures if the true MTBF equals the upper-test MTBF is 2.63.

The expected quantity of test time as a function of true MTBF is:

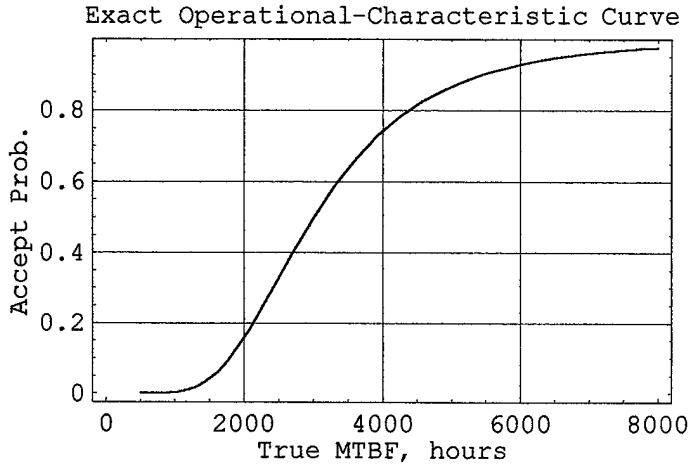
```
Show[expectedtesttimePlot];
```



The expected test time if the true MTBF equals the lower-test MTBF is 9,811 hours. The expected test time if the true MTBF equals the upper-test MTBF is 11,553 hours.

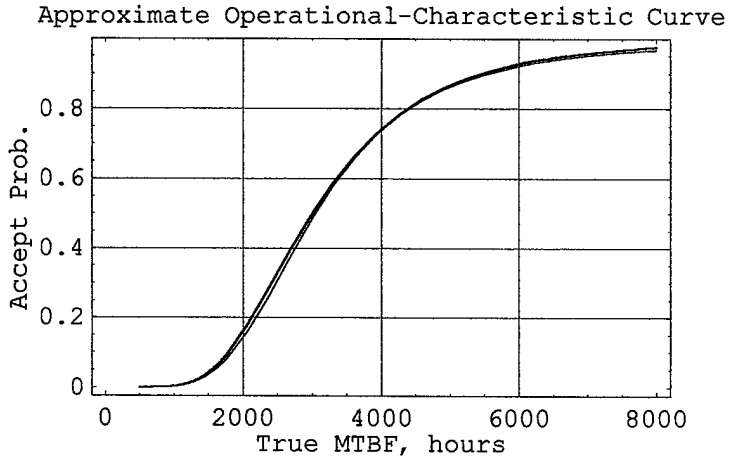
The operational-characteristic curve (i.e., the acceptance probability as a function of true MTBF) is:

```
Show[ocPlot];
```



The exact operational-characteristic curve can be overlaid on the approximate one:

```
Show[approxOC, ocPlot];
```



The approximate operational-characteristic curve was quite good in this case due to the modest level of truncation.

Summary

Test planners and evaluators were provided, in a timely fashion, with a truncated exponential sequential test plan that fairly closely met their requirements. They were also provided with key results of an exact analysis of the test design, including stage-by-stage acceptance, continuation and rejection probabilities, actual consumer and producer risks and operational-characteristic curve.

The sequential test designed in this chapter would likely result in a markedly shorter test length compared with a typical fixed-length test. This chapter can serve as an electronic template for future test designs.

Chapter 5

Simulation-Based Hypergeometric Sequential Test Plan: Case Study 4

Introduction

There are 310 troubleshooting procedures for the maintenance of a new Army vehicle in need of evaluation. Evaluating each of the procedures would be costly in terms of cost and schedule. A statistically-sound method is desired whereby the procedures can be progressively evaluated and, if they are found to be highly error-free, all 310 procedures can be accepted based on a sample. If the procedures are not sufficiently error-free, acceptance based on a sample will not occur therefore all of the procedures will have to be reviewed and corrected.

The hypergeometric distribution, not the binomial, is applicable to this problem since we will be sampling without replacement from a finite population and each troubleshooting procedure will be judged to be acceptable or defective. In contrast to conventional sequential test plans, which ultimately lead to either acceptance or rejection, rejection is not possible with the plan designed in this chapter. If the procedures are not sufficiently error-free, all 310 procedures will be evaluated and subsequently corrected. It doesn't appear that test-design methodology is available in the literature for this case. Consequently, the approach taken here is to develop a hypergeometric sequential simulation and then use it to approximately characterize the behavior of the hypergeometric sequential decision rules. We can through trial-and-error arrive at an acceptable test plan.

Potential Test Plan Decision Rules

We will need functions for the hypergeometric distribution which are defined in the standard add-on package `DiscreteDistributions` which is loaded thus:

```
Needs["Statistics`DiscreteDistributions`"]
```

The usage message for the hypergeometric distribution is:

```
? HypergeometricDistribution

HypergeometricDistribution[n, nsucc, ntot] represents the
  hypergeometric distribution for a sample of size n, drawn without
  replacement from a population with nsucc successes and total
  size ntot.  A HypergeometricDistribution[n, nsucc, ntot] random
  variable describes the number of successes occuring in the sample.
```

It's important to note that random samples must be drawn without replacement at each stage of the test plan!

The entire population of troubleshooting procedures is 310. After some discussion with the test planners and evaluators, it seems prudent to design a test plan where the probability of acceptance based on a sample of the 310 procedures will not appreciably exceed 20% if the number of troubleshooting procedures containing errors is 31 or more.

Let's assume we have 310 troubleshooting procedures 31 of which contain errors (i.e., 90% are error-free). If we draw a sample of 15, the probability of obtaining 0 procedures with errors is:

```
PDF[HypergeometricDistribution[15, 31, 310], 0] // N

0.198032
```

This is the first increment of consumer risk. This value is too high since the test plan should grow stage-by-stage towards 20%, not start there. Let's try a sample of size 20:

```
N[PDF[HypergeometricDistribution[20, 31, 310], 0]]

0.113213
```

This is a more reasonable starting point. It's roughly half of the desired cumulative acceptance probability of 20%. The greatest acceptance probability will occur at the first stage.

Several sequences of decision rules were simulated as described in the rest of this chapter and the following rules were eventually arrived at:

Stage	Procedures to Examine	Cumulative Procedures Examined	Accept if Cumulative Errors ≤
1	20	20	0
2	20	40	1

3	20	60	2
4	20	80	3
5	20	100	4
6	20	120	5
7	20	140	6
8	20	160	7
9	20	180	8
10	20	200	9
11	20	220	10
12	20	240	11
13	20	260	12
14	20	280	13
15	30	310	N/A

Simulations of these rules are provided in subsequent sections of this notebook. Let's graph these accept decision rules. The values for the cumulative number of procedures examined at stages 1 through 14 are:

```
xaxispts = Range[20, 280, 20]
```

```
{20, 40, 60, 80, 100, 120, 140, 160, 180, 200, 220, 240, 260, 280}
```

A list of the decision rules is then:

```
pts = Transpose[{xaxispts, Range[0, 13]}]
```

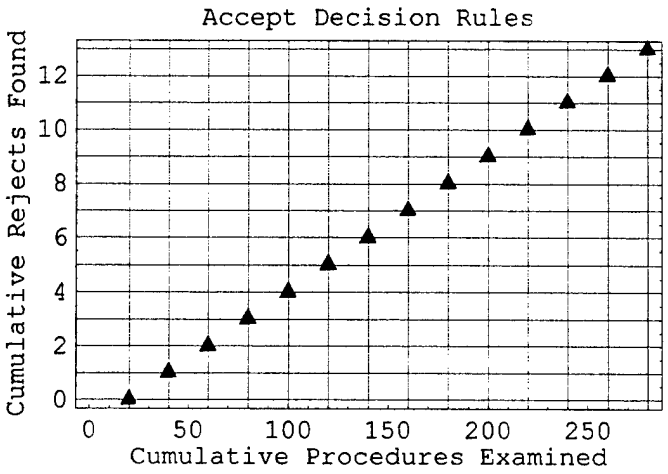
```
{{20, 0}, {40, 1}, {60, 2}, {80, 3}, {100, 4}, {120, 5}, {140, 6}, {160, 7},
 {180, 8}, {200, 9}, {220, 10}, {240, 11}, {260, 12}, {280, 13}}
```

Functions defined in the standard add-on package `MultipleListPlot` will be needed so we'll load the package now.

```
Needs["Graphics`MultipleListPlot`"]
```

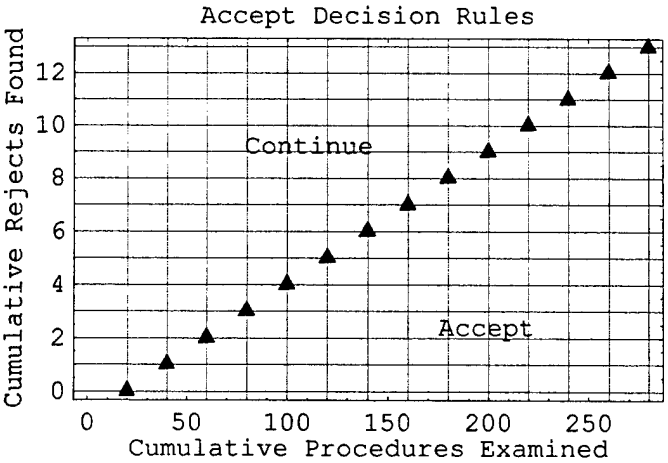
The decision-rule plot is:

```
MultipleListPlot[pts,
  SymbolShape → PlotSymbol[Triangle, 4], Frame → True,
  Axes → False, FrameLabel → {"Cumulative Procedures Examined",
    "Cumulative Rejects Found", "Accept Decision Rules", None},
  GridLines → {xaxispts, Range[0, 13]}];
```



Identification of the reject, continue and accept regions can be overlaid thus:

```
decisionPlot = Show[%, Graphics[{Text["Continue", Scaled[{0.4, 0.7}]],
    Text["Accept", Scaled[{0.7, 0.2}]]}]];
```



It is intended that these decision rules will be used thus:

- Select 20 procedures at random from the population of 310.
- If 0 rejects are found, stop the test and accept all the procedures.
- Otherwise continue to the next stage by randomly selecting 20 procedures from the remaining 290.

- If the cumulative number of rejects equals 1, then stop the test and accept all the procedures.
- Otherwise continue to the next stage by randomly selecting 20 procedures from the remaining 270.

⋮

The decision rules are defined for use in simulations later in this notebook.

```
accept[0] = -1;
```

```
Do[accept[i] = i - 1, {i, 1, 15 - 1}]
```

Here are the rules:

```
? accept
```

```
Global`accept

accept[0] = -1

accept[1] = 0

accept[2] = 1

accept[3] = 2

accept[4] = 3

accept[5] = 4

accept[6] = 5

accept[7] = 6

accept[8] = 7

accept[9] = 8

accept[10] = 9

accept[11] = 10

accept[12] = 11

accept[13] = 12

accept[14] = 13
```

Development of Simulation for Potential Decision Rules

In this section, we develop functions needed for simulating hypergeometric sequential test plans. In the next section these functions will be used to simulate and approximately characterize the decision rules proposed herein.

The samples that will be taken from the original population of 310 (unless acceptance occurs) are assigned as the value of the symbol *sample*:

```
sample = Append[Table[20, {14}], 30]
```

```
{20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 30}
```

The quantity of samples is:

```
Length[sample]
```

```
15
```

The quantities of unexamined procedures at stages 0 through 15 are assigned as the value of the symbol *proc*:

```
proc = 310 - FoldList[Plus, 0, sample]
```

```
{310, 290, 270, 250, 230, 210, 190, 170, 150, 130, 110, 90, 70, 50, 30, 0}
```

The number of such quantities is:

```
Length[proc]
```

```
16
```

The initial quantity of defective procedures is assigned as the value of the symbol *initdef*:

```
initdef = 31;
```

Next we will define a function that will increment the stage number by one and add a hypergeometric random variable to the running total of defective procedures discovered during the test.

```
fun[{stage_Integer, cumdef_Integer}] :=
 {stage + 1, cumdef + Random[HypergeometricDistribution[
     sample[[stage+1]], initdef - cumdef, proc[[stage+1]]]]}
```

Now we will define a function that will be used to test the simulation against the acceptance rules. The simulation will continue as long as the running total of defective procedures exceeds the acceptable quantity for that stage.

```
testfun[{stage_Integer, cumdef_Integer}] := accept[stage] < cumdef
```

Let's reset the pseudorandom number generator using the integer one as a seed. This will allow a repeatable result which is helpful when de-bugging code.

```
SeedRandom[1]
```

Now we can simulate a single hypergeometric sequential test plan as follows:

```
NestWhileList[fun, {0, 0}, testfun, 1, Length[sample]]
```

```
{{0, 0}, {1, 2}, {2, 5}, {3, 8}, {4, 8}, {5, 12}, {6, 14}, {7, 17}, {8, 21},
 {9, 21}, {10, 23}, {11, 25}, {12, 27}, {13, 28}, {14, 28}, {15, 31}}
```

This simulation went the distance. After some experimentation, it was found that using the integer 10 as the seed results in a stage-one acceptance:

```
SeedRandom[10]
```

```
NestWhileList[fun, {0, 0}, testfun, 1, Length[sample]]
```

```
{{0, 0}, {1, 0}}
```

In the next section, much larger simulations will be run using the functions just defined.

Simulation of Decision Rules

- **Assume Defect Quantity Equals 40**

If 40 of the 310 procedures are defective, then the percentage of defectives is

```
40
--- // N
310
```

```
0.129032
```

and the percentage of non-defectives is

```
1 - %
```

```
0.870968
```

The initial quantity of defective procedures is assigned as the value of the symbol *initdef*:

```
initdef = 40;
```

The pseudorandom number generating function is:

```
fun[{stage_Integer, cumdef_Integer}] :=
 {stage + 1, cumdef + Random[HypergeometricDistribution[
     sample[[stage+1]], initdef - cumdef, proc[[stage+1]]]]}
```

The function that will test the simulation against the acceptance rules is:

```
testfun[{stage_Integer, cumdef_Integer}] := accept[stage] < cumdef
```

The desired quantity of simulated hypergeometric sequential tests is assigned as the value of the symbol *simqty*:

```
simqty = 25000;
```

A simulation of 25,000 hypergeometric tests is generated as follows:

```
simlist = Table[
   Length[NestWhileList[fun, {0, 0}, testfun, 1, Length[sample]]] - 1,
   {simqty}];
```

The symbol *simlist* contains a list of the stages that the simulations ended at. The quantity of terminations at each stage are assigned as the value of *simlist40*:

```
simlist40 =
  Table[Length[Select[simlist, #1 == i &]], {i, 1, Length[sample]}];
```

The filename *hypersimfile3* will be assigned as the value of the symbol *simfile*.

```
simfile = "hypersimfile3";
```

The simulation results are saved to this file so that they can be retrieved for subsequent analysis if need be.

```
Save[simfile, simlist40]
```

These simulation results may be easily retrieved thus:

```
<< "hypersimfile3";
```

Stage-by-stage termination probabilities are:

```
TableForm[Transpose[{Range[1, 15], N[simlist40/simqty]}],
 TableHeadings → {None, {"Stage", "Termination Probability"}},
 TableAlignments → Center]
```

Stage	Termination Probability
1	0.05824
2	0.00896
3	0.00228
4	0.0004
5	0.00004
6	0.00008
7	0.
8	0.
9	0.
10	0.
11	0.
12	0.
13	0.
14	0.
15	0.93

The terminations at stages 1 - 14 are acceptances. The terminations at the last stage are simulations that went the distance. Cumulative termination probabilities are:

```
TableForm[Transpose[{Range[0, 15], N[FoldList[Plus, 0, simlist40/simqty]]}],
 TableHeadings →
  {None, {"Stage", "Cumulative Termination Probability"}},
 TableAlignments → Center]
```

Stage	Cumulative Termination Probability
0	0.
1	0.05824
2	0.0672
3	0.06948
4	0.06988
5	0.06992
6	0.07
7	0.07
8	0.07
9	0.07
10	0.07
11	0.07
12	0.07
13	0.07
14	0.07
15	1.

The stage 1 acceptance probability is:

```
PDF[HypergeometricDistribution[20, 40, 310], 0] // N
```

```
0.0573767
```

The stage 1 acceptance probability obtained by simulation is consistent with the numerical result above. The final cumulative acceptance probability is the stage 14 termination probability, i.e., approximately 7%.

- **Simulation of Additional Defect Quantities**

Appendix E contains the bulk of the simulation results for the hypergeometric sequential test plan designed in this chapter and was executed in conjunction with it. Appendix E contains simulations assuming the number of defective procedures in the population of 310 was 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34 and 37.

Stage-by-Stage Acceptance Probabilities

In this section, stage-by-stage acceptance probabilities will be plotted.

First let's retrieve the simulation results:

```
<< "hypersimfile3";
```

Lists of the cumulative acceptance points for each case simulated will be obtained next:

```
cumacc40 = N[Delete[FoldList[Plus, 0, simlist40/simqty], {{1}, {16}}]]
```

```
{0.05824, 0.0672, 0.06948, 0.06988, 0.06992,
 0.07, 0.07, 0.07, 0.07, 0.07, 0.07, 0.07, 0.07, 0.07}
```

```
cumacc37 = N[Delete[FoldList[Plus, 0, simlist37/simqty], {{1}, {16}}]]
```

```
{0.07104, 0.08424, 0.08796, 0.08892, 0.08912, 0.08916,
 0.08916, 0.0892, 0.0892, 0.0892, 0.0892, 0.0892, 0.0892, 0.0892}
```

```
cumacc34 = N[Delete[FoldList[Plus, 0, simlist34/simqty], {{1}, {16}}]]
```

```
{0.09204, 0.111, 0.11652, 0.11896, 0.11964, 0.12, 0.12008,
 0.12008, 0.12008, 0.12008, 0.12008, 0.12008, 0.12008, 0.12008}
```

```
cumacc31 = N[Delete[FoldList[Plus, 0, simlist31/simqty], {{1}, {16}}]]
```

```
{0.115, 0.1462, 0.156, 0.16008, 0.16152, 0.16188, 0.16208,
 0.16212, 0.16212, 0.16212, 0.16212, 0.16212, 0.16212, 0.16212}
```

```
cumacc28 = N[Delete[FoldList[Plus, 0, simlist28/simqty], {{1}, {16}}]]
```

```
{0.14216, 0.18124, 0.1954, 0.202, 0.20524, 0.20636, 0.20696,
 0.2074, 0.2074, 0.2074, 0.2074, 0.2074, 0.2074, 0.2074}
```

$$\text{cumacc25} = \text{N}\left[\text{Delete}\left[\text{FoldList}\left[\text{Plus}, 0, \frac{\text{simlist25}}{\text{simqty}}\right], \{\{1\}, \{16\}\}\right]\right]$$

{0.17484, 0.2296, 0.2542, 0.26608, 0.27372, 0.27872, 0.28084,
0.28168, 0.28228, 0.2824, 0.28244, 0.28244, 0.28244, 0.28244}

$$\text{cumacc22} = \text{N}\left[\text{Delete}\left[\text{FoldList}\left[\text{Plus}, 0, \frac{\text{simlist22}}{\text{simqty}}\right], \{\{1\}, \{16\}\}\right]\right]$$

{0.22056, 0.29716, 0.33464, 0.35728, 0.3714, 0.38032, 0.38584,
0.38948, 0.3918, 0.39264, 0.39296, 0.39308, 0.39316, 0.39316}

$$\text{cumacc19} = \text{N}\left[\text{Delete}\left[\text{FoldList}\left[\text{Plus}, 0, \frac{\text{simlist19}}{\text{simqty}}\right], \{\{1\}, \{16\}\}\right]\right]$$

{0.26868, 0.37168, 0.42664, 0.46164, 0.48568, 0.50348, 0.51712,
0.5268, 0.53392, 0.53944, 0.5428, 0.54484, 0.546, 0.54632}

$$\text{cumacc16} = \text{N}\left[\text{Delete}\left[\text{FoldList}\left[\text{Plus}, 0, \frac{\text{simlist16}}{\text{simqty}}\right], \{\{1\}, \{16\}\}\right]\right]$$

{0.33352, 0.46376, 0.53612, 0.58772, 0.62432, 0.65468, 0.68032,
0.7004, 0.71812, 0.73352, 0.74772, 0.76104, 0.77324, 0.7828}

$$\text{cumacc13} = \text{N}\left[\text{Delete}\left[\text{FoldList}\left[\text{Plus}, 0, \frac{\text{simlist13}}{\text{simqty}}\right], \{\{1\}, \{16\}\}\right]\right]$$

{0.41144, 0.575, 0.66796, 0.73096, 0.77728, 0.81316, 0.84472,
0.8718, 0.89688, 0.91956, 0.94104, 0.96092, 0.98216, 1.}

$$\text{cumacc10} = \text{N}\left[\text{Delete}\left[\text{FoldList}\left[\text{Plus}, 0, \frac{\text{simlist10}}{\text{simqty}}\right], \{\{1\}, \{16\}\}\right]\right]$$

{0.50552, 0.701, 0.8024, 0.86652, 0.90868,
0.94004, 0.9636, 0.97984, 0.99068, 0.9974, 1., 1., 1., 1.}

$$\text{cumacc7} = \text{N}\left[\text{Delete}\left[\text{FoldList}\left[\text{Plus}, 0, \frac{\text{simlist7}}{\text{simqty}}\right], \{\{1\}, \{16\}\}\right]\right]$$

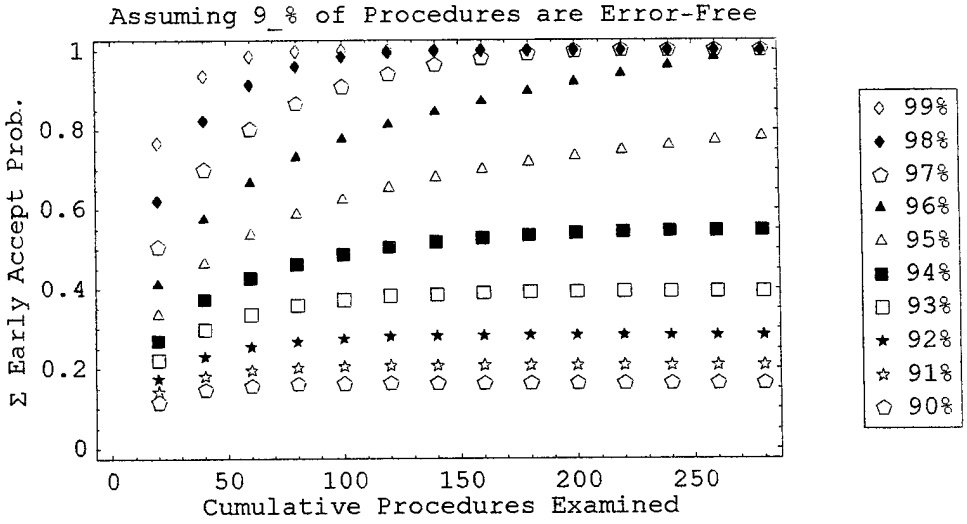
{0.6214, 0.82416, 0.91408, 0.95992,
0.98368, 0.9944, 0.999, 1., 1., 1., 1., 1., 1., 1.}

```
cumacc4 = N[Delete[FoldList[Plus, 0, simlist4/simqty], {{1}, {16}}]]
```

```
{0.7682, 0.93672, 0.9854, 0.99832, 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.}
```

We'll plot just the cumulative acceptance probabilities for the cases where the true fraction of error-free procedures was 90% or higher.

```
MultipleListPlot[Transpose[{xaxispts, cumacc4}],
  Transpose[{xaxispts, cumacc7}], Transpose[{xaxispts, cumacc10}],
  Transpose[{xaxispts, cumacc13}],
  Transpose[{xaxispts, cumacc16}], Transpose[{xaxispts, cumacc19}],
  Transpose[{xaxispts, cumacc22}], Transpose[{xaxispts, cumacc25}],
  Transpose[{xaxispts, cumacc28}], Transpose[{xaxispts, cumacc31}],
  Axes → False, Frame → True, FrameLabel →
   {"Cumulative Procedures Examined", "Σ Early Accept Prob.",
    "Assuming 9_% of Procedures are Error-Free", None},
  PlotLegend → {"99%", "98%", "97%", "96%", "95%", "94%",
    "93%", "92%", "91%", "90%"}, SymbolShape →
   {PlotSymbol[Diamond, Filled → False], PlotSymbol[Diamond,
     Filled → True], MakeSymbol[RegularPolygon[5, 3]], PlotSymbol[
     Triangle, Filled → True], PlotSymbol[Triangle, Filled → False],
    PlotSymbol[Box, Filled → True], PlotSymbol[Box, Filled → False],
    PlotSymbol[Star, Filled → True], PlotSymbol[Star, Filled → False],
    MakeSymbol[RegularPolygon[5, 3]]}];
```



We'll also tabulate the cumulative acceptance probabilities for all cases simulated:

```
TableForm[
 Prepend[Transpose[{Range[1, 14], xaxispts, Round[100 cumacc40],
    Round[100 cumacc37], Round[100 cumacc34], Round[100 cumacc31],
    Round[100 cumacc28], Round[100 cumacc25], Round[100 cumacc22],
    Round[100 cumacc19], Round[100 cumacc16], Round[100 cumacc13],
    Round[100 cumacc10], Round[100 cumacc7], Round[100 cumacc4]}],
  {"Stage", "Σ Proc.", "40", "37", "34", "31", "28", "25",
   "22", "19", "16", "13", "10", "7", "4"}],
 TableHeadings → None, TableDirections → {Column, Row},
 TableAlignments → Center, TableSpacing → 1.3]
```

Stage	Σ Proc.	40	37	34	31	28	25	22	19	16	13	10	7	4
1	20	6	7	9	12	14	17	22	27	33	41	51	62	77
2	40	7	8	11	15	18	23	30	37	46	57	70	82	94
3	60	7	9	12	16	20	25	33	43	54	67	80	91	99
4	80	7	9	12	16	20	27	36	46	59	73	87	96	100
5	100	7	9	12	16	21	27	37	49	62	78	91	98	100
6	120	7	9	12	16	21	28	38	50	65	81	94	99	100
7	140	7	9	12	16	21	28	39	52	68	84	96	100	100
8	160	7	9	12	16	21	28	39	53	70	87	98	100	100
9	180	7	9	12	16	21	28	39	53	72	90	99	100	100
10	200	7	9	12	16	21	28	39	54	73	92	100	100	100
11	220	7	9	12	16	21	28	39	54	75	94	100	100	100
12	240	7	9	12	16	21	28	39	54	76	96	100	100	100
13	260	7	9	12	16	21	28	39	55	77	98	100	100	100
14	280	7	9	12	16	21	28	39	55	78	100	100	100	100

Operational-Characteristic Curve

In this section, an approximate operational-characteristic curve will be generated. First, the points are:

```
N[ocpts = {{1 - 40/310, Last[cumacc40]},
    {1 - 37/310, Last[cumacc37]}, {1 - 34/310, Last[cumacc34]},
    {1 - 31/310, Last[cumacc31]}, {1 - 28/310, Last[cumacc28]},
    {1 - 25/310, Last[cumacc25]}, {1 - 22/310, Last[cumacc22]},
    {1 - 19/310, Last[cumacc19]}, {1 - 16/310, Last[cumacc16]},
    {1 - 13/310, Last[cumacc13]}, {1 - 10/310, Last[cumacc10]},
    {1 - 7/310, Last[cumacc7]}, {1 - 4/310, Last[cumacc4]}}]
```

```
{{0.870968, 0.07}, {0.880645, 0.0892}, {0.890323, 0.12008},
 {0.9, 0.16212}, {0.909677, 0.2074}, {0.919355, 0.28244},
 {0.929032, 0.39316}, {0.93871, 0.54632}, {0.948387, 0.7828},
 {0.958065, 1.}, {0.967742, 1.}, {0.977419, 1.}, {0.987097, 1.}}
```

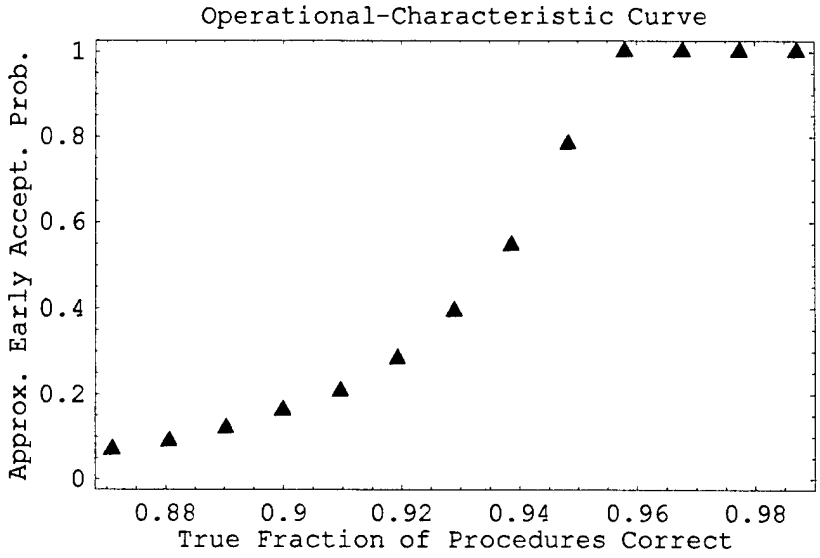
The points can be formatted in a table thus:

```
TableForm[N[ocpts],
 TableHeadings → {None, {"True Fraction Correct Procedures",
    "Acceptance Probability"}}, TableAlignments → Center]
```

True Fraction Correct Procedures	Acceptance Probability
0.870968	0.07
0.880645	0.0892
0.890323	0.12008
0.9	0.16212
0.909677	0.2074
0.919355	0.28244
0.929032	0.39316
0.93871	0.54632
0.948387	0.7828
0.958065	1.
0.967742	1.
0.977419	1.
0.987097	1.

Now an approximate operational-characteristic curve is generated:

```
MultipleListPlot[ocpts,
  SymbolShape → PlotSymbol[Triangle, 4], Frame → True, Axes → False,
  FrameLabel → {"True Fraction of Procedures Correct",
    "Approx. Early Accept. Prob.",
    "Operational-Characteristic Curve", None}];
```



If 90% or fewer of the 310 procedures are correct, the probability of acceptance based on a sample will not appreciably exceed 16%. If 95% or more of the procedures are correct, the probability of acceptance based on a sample exceeds 75%. And if 96, 97 or 98% of the procedures are correct, its likely that acceptance will occur by stage 3 (60 procedures examined), stage 2 (40 procedures examined) or stage 1 (20 procedures examined), respectively.

Summary

A hypergeometric sequential test plan was proposed and examined in this notebook. Simulations were performed assuming the number of defective procedures in the population of 310 was 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34, 37 and 40. An approximate operational-characteristic curve was plotted. Approximate stage-by-stage acceptance probabilities were obtained and plotted as well. The test plan designed in this chapter is very good at not accepting troubleshooting procedures (based on sampling) when the percentage of correct procedures is less than 90%. The plan is also very good at acceptance if the true percentage of correct procedures is greater than or equal to 95%. And if the true percentage of correct procedures is greater than or equal to 96%, acceptance should occur quickly.

This chapter documents a new, simulation-based hypergeometric sequential test design for evaluating maintenance or troubleshooting procedures. The method was developed for a new Army vehicle. The quantity of troubleshooting procedures was large enough that evaluation of all procedures would be costly in terms of cost and schedule. Yet the quantity of procedures was too small to use binomial sequential test methods available in the literature. A statistically-sound method was developed whereby the procedures can be progressively evaluated and, if they are found to be highly error-free, all procedures can be accepted based on a sample. If the procedures are not sufficiently error-free, acceptance based on a sample will not occur. Thus the new test-design method provides for accept-continue decision-making, not the traditional accept-continue-reject decision-making.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 6

Summary

Sequential test plans can be highly beneficial because expected test lengths and sample sizes can be often reduced by up to half compared with the more common fixed-length test plans. The design and analysis of sequential test plans is quite challenging. The purpose of this report is to document and disseminate recent improvements in order to help the Army take advantage of sequential test plans while avoiding the pitfalls.

A key accomplishment included in this report concerns the exact-analysis method for exponential sequential test designs. Previously, such exact-analysis methodology was for all practical purposes restricted to the statistical research community. Indeed, little practical use was found for these methods for the past forty years. It was possible to re-formulate and implement the exact-analysis method in modern mathematics software in a form that can, for the first time, be readily used by test planners. It was deemed decisively advantageous to undertake this effort because of the resurgence of truncated exponential sequential test designs, the properties of which are very difficult to obtain otherwise.

Chapter 2 contains a case study of a contractor-proposed exponential sequential test design for an imaging system. The test design was both analyzed exactly and simulated in order to rapidly characterize it, including assessing the impact of truncation. The test planners were then able to confidently accept the plan. This chapter can serve as a template for the verification of such test plans. Indeed, the author has already had occasion to do so many times.

Chapter 3 contains a case study that illustrates a critical but not infrequent error contained in a recent, contractor-proposed exponential sequential test design for another imaging system. The proposed test plan would have resulted in a risk to the Army of 41%, approximately twice the Army's not-to-exceed value of 20%. This chapter was prepared in order to clearly illustrate this mistake and thereby help test planners avoid it in the future.

Chapter 4 contains the design and exact analysis of a truncated, exponential sequential test plan for a tactical terminal. The sequential test design would likely result in a markedly shorter test length compared with a typical fixed-length test. A test design that met the program's requirement was provided in a timely fashion. This chapter can serve as template for future test designs.

Chapter 5 documents a new, simulation-based hypergeometric sequential test design for evaluating maintenance or troubleshooting procedures. The method was developed for a new Army vehicle where the quantity of troubleshooting procedures was large enough that evaluation of all procedures would be costly in terms of cost and schedule. Yet the quantity of procedures was too small to use binomial sequential test methods available in the literature. A statistically-sound method was developed whereby the procedures can be progressively evaluated and, if they are found to be highly error-free, all procedures can be accepted based on a sequence of samples. If the procedures are not sufficiently error-free, acceptance based on sampling should not occur. Thus the new test-design method provides for accept-continue decision-making, not the traditional accept-continue-reject decision-making.

References

Butler, D.A., Lieberman, G.J., "An early-accept modification to the test plans of Military Standard 781C", Technical Report 196, Departments of Operations Research and Statistics, Stanford University, 1980.

Epstein, B., "Truncated life tests in the exponential case", *Annals of Mathematical Statistics*, vol. 25, pp. 555-564, 1954.

Epstein, B., Patterson, A., Qualls, C., "The exact analysis of sequential life tests with particular application to AGREE plans", *Proc. Joint AIAA-SAE-ASME Aerospace Reliability and Maintainability Conference*, May 1963, pp. 284-311.

Epstein, B., Sobel, M., "Sequential life tests in the exponential case", *Annals of Mathematical Statistics*, vol. 26, pp. 82-93, 1955.

Kececioglu, D., *Reliability & Life Testing Handbook*, vol. 1, Prentice Hall, Upper Saddle River, NJ, 1993, pp. 198-207.

McCullough, B.D., "The accuracy of *Mathematica* 4 as a statistical package", *Computational Statistics 15*, pp. 279-299, 2000.

Wolfram, S., *The Mathematica Book*, Wolfram Media/Cambridge University Press, 4th ed., 1999.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix A

Exact Analysis of Proposed Exponential Sequential Decision Rules from Chapter 2

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix A

Exact Analysis of Proposed Exponential Sequential Decision Rules from Chapter 2

Introduction

An exact method was developed by Epstein, Patterson and Qualls [1963] for analyzing a sequence of decision rules such as the one simulated in chapter 2. This appendix contains an analysis of the exact stage-by-stage acceptance, continuation and rejection probabilities resulting from the sequence of decision rules. Included are the important special cases that arise at the last stage: consumer risk and operational-characteristic curve. *Mathematica* symbolics are used to obtain results with the true Mean Time Between Failures (MTBF) held symbolic until a numerical value is supplied. The stage-by-stage calculations are performed in such a way that numerical errors that would otherwise accumulate are entirely avoided. The results of all calculations are "exact" but include occurrences of the exponential function. Numerical approximations to any desired precision are provided as well.

Setup

Functions contained in the standard add-on package `Statistics`DiscreteDistributions`` are needed by this method which we load now:

```
Needs["Statistics`DiscreteDistributions`"]
```

Formulate Reliability Test Plan Decision Rules

In order to apply the exact-analysis method, we need to construct a list of accept points from the decision rules provided in chapter 2. Each pair will be of the form $\{t_i, i\}$ where the first pair defines the zero-failure accept time, the second pair defines the one-failure accept time, etc. The accept rules are assigned as the value of the symbol *accept*:

```
N[accept = {{1395, 0}, {2092, 1},
   {2789, 2}, {3487, 3}, {4184, 4}, {4881, 5}, {5578, 6}}]
```

```
{{1395., 0.}, {2092., 1.}, {2789., 2.},
 {3487., 3.}, {4184., 4.}, {4881., 5.}, {5578., 6.}}
```

We need to construct a list of reject points from these decision rules. Each pair will be of the form $\{t_i, i\}$ where the first pair defines the shortest reject time and the corresponding quantity of failures, the second defines the second-shortest reject time and the corresponding quantity of failures, etc. The reject rules are assigned as the value of the symbol *reject*:

```
N[reject = {{697, 3}, {1395, 4}, {2092, 5}, {2789, 6}, {5578, 7}}]
```

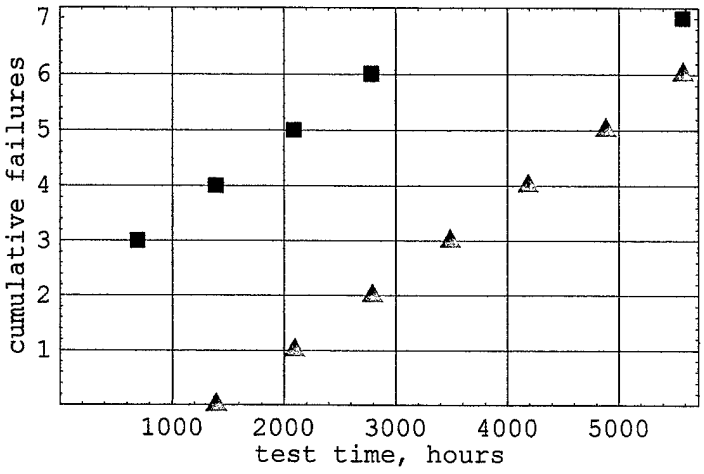
```
{{697., 3.}, {1395., 4.}, {2092., 5.}, {2789., 6.}, {5578., 7.}}
```

It would be helpful to graphically depict the decision rules for this test design. We will need functions contained in the standard add-on package `Graphics`MultipleListPlot`` which we load now:

```
Needs["Graphics`MultipleListPlot`"]
```

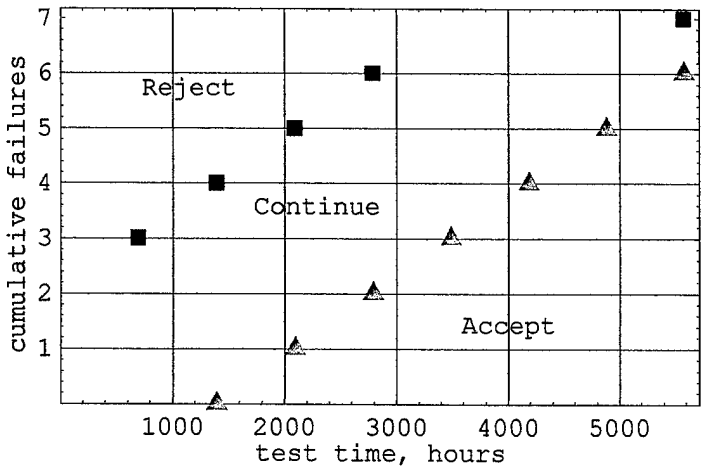
The decision rules, with the accept and reject points represented by triangles and boxes, respectively, are plotted as follows:

```
decisionPlot =
  MultipleListPlot[accept, Reverse[reject], PlotJoined → False,
   PlotRange → {{0, Automatic}, {0, Automatic}}, Frame → True,
   FrameLabel → {"test time, hours", "cumulative failures"},
   GridLines → Automatic,
   SymbolShape → {PlotSymbol[Triangle, 5], PlotSymbol[Box, 3]},
   SymbolStyle → {RGBColor[0, 1, 0], RGBColor[1, 0, 0]}];
```



Identification of the reject, continue and accept regions can be overlaid thus:

```
Show[%, Graphics[{Text["Reject", Scaled[{0.2, 0.8}]], Text["Continue",
    Scaled[{0.4, 0.5}]], Text["Accept", Scaled[{0.7, 0.2}]]}]];
```



Define Function for Stage Times

In this step, we will construct a stage-time function. First, a list is needed of the times for each stage. The stage times are comprised of the accept and reject times joined into a single list and sorted from shortest to longest. The list of stage times is constructed as follows:

```
timeValues =
 Sort[Union[First[Transpose[accept]], First[Transpose[reject]]], Less]
```

```
{697, 1395, 2092, 2789, 3487, 4184, 4881, 5578}
```

It should be noted that the times are expressed as exact numbers (i.e., either as integers or rational numbers) in order to avoid approximations until after the stage-by-stage calculations are complete. If the times are expressed in decimal form, *Mathematica* will treat them as approximate and will use machine-precision (unless many zeroes are used).

It should also be noted that the function `Union` was used to eliminate any repeats occurring as the two lists were combined.

The quantity of stages is:

```
Length[timeValues]
```

```
8
```

A function which will provide time values as a function of stage, except for the special case of stage zero, is:

```
t[stage_Integer /; stage > 0] := timeValues[[stage]]
```

The initial condition for time [Epstein, et al. 1963, equation 16]:

```
t[i_ /; i == 0] := 0
```

Construct Accept-Number Function

In this step, we will construct an accept-number function. First, we will generate an `InterpolatingFunction` object from *accept*:

```
fA = Interpolation[accept, InterpolationOrder → 1]

InterpolatingFunction[{{1395, 5578}}, <>]
```

Now, we define a function which will provide an integer-valued accept number for each stage using Epstein, et al. 1963, equation 11:

```
a[stage_Integer /; stage > 0] := -1 /; t[stage] < First[First[accept]]

a[stage_Integer /; stage > 0] := Floor[fA[t[stage]]]
```

A special case of the accept-number function is defined for the initial condition at stage zero [Epstein, et al. 1963, equation 16]:

```
a[stage_Integer /; stage == 0] := -1
```

Construct Reject-Number Function

In this step, we will construct a reject-number function. First, we will generate an `InterpolatingFunction` object from *reject*:

```
fR = Interpolation[reject, InterpolationOrder → 1]

InterpolatingFunction[{{697, 5578}}, <>]
```

Now, we define an function which will provide an integer-valued reject number for each stage using Epstein, et al. 1963, equation 12:

```
r[stage_Integer /; stage > 0] := Ceiling[fR[t[stage]]]
```

A special case of the reject-number function is defined for the initial condition at stage zero:

```
r[stage_ /; stage == 0] := 1
```

Tabulation of Accept, Continuation and Reject Points

In this step, we generate a table of accept, continuation and reject numbers. This is done to provide a convenient stage-by-stage listing of the test plan to be analyzed. The table is generated as follows:

```
TableForm[Transpose[{Range[Length[timeValues]],
   Table[N[t[stage]], {stage, 1, Length[timeValues]}],
   Table[a[stage], {stage, 1, Length[timeValues]}],
   Append[Table[a[stage] + 1, {stage, 1, Length[timeValues] - 1}], NA],
   Append[Table[r[stage] - 1, {stage, 1, Length[timeValues] - 1}], NA],
   Table[r[stage], {stage, 1, Length[timeValues]}]}],
 TableHeadings -> {None, {"Stage", "Time", "Accept",
    "Continue (min)", "Continue (max)", "Reject"}},
 TableSpacing → {1, 1.5}, TableAlignments → Center]
```

Stage	Time	Accept	Continue (min)	Continue (max)	Reject
1	697.	-1	0	2	3
2	1395.	0	1	3	4
3	2092.	1	2	4	5
4	2789.	2	3	5	6
5	3487.	3	4	6	7
6	4184.	4	5	6	7
7	4881.	5	6	6	7
8	5578.	6	NA	NA	7

Construct Function for Acceptance/Continuation Probability for a Quantity of Failures

In this step, we construct a function for calculating acceptance/continuation probabilities for a quantity of failures [Epstein, et al. 1963, equation 17]:

```
ACProbability[stage_, failure_, trueMTBF_] /;
  And[stage > 0, (a[stage - 1] + 1) ≤ failure ≤ r[stage] - 1] :=
 aclist[stage, failure, trueMTBF]

ACProbability[stage_, failure_, trueMTBF_] /;
  And[stage > 0, Not[(a[stage - 1] + 1) ≤ failure ≤ r[stage] - 1]] := 0
```

Two initial conditions for this function are also needed [Epstein, et al. 1963, equation 16]:

```
ACProbability[0, 0, trueMTBF_] := 1

ACProbability[0, failure_Integer /; failure > 0, trueMTBF_] := 0
```

Up-front Calculation of Acceptance/Continuation Probabilities

In order to reduce execution time, stage-by-stage calculations of acceptance and continuation probabilities are developed in this step.

A function for building up the calculations is:

```
aclistfunction[stage_Integer, failure_Integer, trueMTBF_] :=
 aclist[stage, failure, trueMTBF] =
    Sum[ACProbability[stage - 1, j, trueMTBF]
      PDF[PoissonDistribution[(t[stage] - t[stage - 1])/trueMTBF], failure - j],
     {j, a[stage-1]+1, failure}]
```

An indexed variable *aclist* is used to build up the acceptance and continuation probabilities.

The acceptance and continuation points for the stages are:

```
Map[aclistfunction[1, #, trueMTBF] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 1]]
```

$$\left\{e^{-697/\text{trueMTBF}},\ \frac{697\ e^{-697/\text{trueMTBF}}}{\text{trueMTBF}},\ \frac{485809\ e^{-697/\text{trueMTBF}}}{2\ \text{trueMTBF}^2}\right\}$$

```
Map[aclistfunction[2, #, trueMTBF] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 2]]
```

$$\left\{e^{-1395/\text{trueMTBF}},\ \frac{1395\ e^{-1395/\text{trueMTBF}}}{\text{trueMTBF}},\ \frac{1946025\ e^{-1395/\text{trueMTBF}}}{2\ \text{trueMTBF}^2},\ \frac{1188048001\ e^{-1395/\text{trueMTBF}}}{3\ \text{trueMTBF}^3}\right\}$$

```
Map[aclistfunction[3, #, trueMTBF] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 3]]
```

$$\left\{\frac{1395\ e^{-2092/\text{trueMTBF}}}{\text{trueMTBF}},\ \frac{3890655\ e^{-2092/\text{trueMTBF}}}{2\ \text{trueMTBF}^2},\ \frac{4239172471\ e^{-2092/\text{trueMTBF}}}{3\ \text{trueMTBF}^3},\ \frac{7093185960133\ e^{-2092/\text{trueMTBF}}}{12\ \text{trueMTBF}^4}\right\}$$

```
Map[aclistfunction[4, #, trueMTBF] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 4]]
```

$$\left\{\frac{3890655\, e^{-2789/\text{trueMTBF}}}{2\,\text{trueMTBF}^2},\ \frac{16613704547\, e^{-2789/\text{trueMTBF}}}{6\,\text{trueMTBF}^3},\ \frac{12291172226983\, e^{-2789/\text{trueMTBF}}}{6\,\text{trueMTBF}^4},\ \frac{1730036199487099\, e^{-2789/\text{trueMTBF}}}{2\,\text{trueMTBF}^5}\right\}$$

```
Map[aclistfunction[5, #, trueMTBF] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 5]]
```

$$\left\{\frac{16613704547\, e^{-3487/\text{trueMTBF}}}{6\,\text{trueMTBF}^3},\ \frac{23887538000789\, e^{-3487/\text{trueMTBF}}}{6\,\text{trueMTBF}^4},\ \frac{5938826155984575\, e^{-3487/\text{trueMTBF}}}{2\,\text{trueMTBF}^5},\ \frac{11337723855460591714\, e^{-3487/\text{trueMTBF}}}{9\,\text{trueMTBF}^6}\right\}$$

```
Map[aclistfunction[6, #, trueMTBF] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 6]]
```

$$\left\{\frac{23887538000789\, e^{-4184/\text{trueMTBF}}}{6\,\text{trueMTBF}^4},\ \frac{17233046227251829\, e^{-4184/\text{trueMTBF}}}{3\,\text{trueMTBF}^5},\ \frac{154673751220700754709\, e^{-4184/\text{trueMTBF}}}{36\,\text{trueMTBF}^6}\right\}$$

```
Map[aclistfunction[7, #, trueMTBF] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 7]]
```

$$\left\{\frac{17233046227251829\, e^{-4881/\text{trueMTBF}}}{3\,\text{trueMTBF}^5},\ \frac{298810949865435052465\, e^{-4881/\text{trueMTBF}}}{36\,\text{trueMTBF}^6}\right\}$$

```
Map[aclistfunction[8, #, trueMTBF] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 8]]
```

$$\left\{\frac{298810949865435052465\, e^{-5578/\text{trueMTBF}}}{36\,\text{trueMTBF}^6}\right\}$$

Construct Function for Acceptance Probability for a Quantity of Failures

In this step, we construct a function for calculating acceptance probabilities for a quantity of failures [Epstein, et al. 1963, equation 18]:

```
AcceptanceProbability[stage_Integer, failure_Integer, trueMTBF_] :=
 ACProbability[stage, failure, trueMTBF]
```

Construct and Use Function for Acceptance Probability for Each Stage

In this step, we construct and use a function for calculating stage-by-stage acceptance probabilities [Epstein, et al. 1963, equation 20]:

```
AcceptanceProbability[stage_Integer, trueMTBF_] :=
  Sum[AcceptanceProbability[stage, failure, trueMTBF], {failure, a[stage-1]+1, a[stage]}] /;
   a[stage - 1] < a[stage]

AcceptanceProbability[stage_Integer, trueMTBF_] :=
 0 /; Not[a[stage - 1] < a[stage]]
```

The acceptance probability as a function of true MTBF is the sum of the probabilities of acceptance at each stage. This is given by Epstein, et al. 1963, equation 14:

```
AcceptanceProbability[trueMTBF_] :=
 Sum[AcceptanceProbability[trueMTBF_, i_], {i, 1, n}]
```

- **Cumulative Acceptance Probabilities for Each Stage When *trueMTBF* is Symbolic**

The cumulative acceptance probability for stage one when *trueMTBF* is left symbolic is:

```
Sum[AcceptanceProbability[stage, trueMTBF], {stage, 1, 1}]
```

```
0
```

This result is obviously correct since the first opportunity for acceptance to occur is at stage two. The cumulative acceptance probability for stage two when *trueMTBF* is left symbolic is:

```
Sum[AcceptanceProbability[stage, trueMTBF], {stage, 1, 2}]
```

$$e^{-1395/\text{trueMTBF}}$$

The result above is exact but partially symbolic. An exact result can be obtained for a specific value of *trueMTBF* such as the lower-test MTBF as follows:

```
% /. trueMTBF → 503
```

$$\frac{1}{e^{1395/503}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.0624518236575710360232 5
```

An answer accurate to 22 decimal places was elicited not because an answer this precise was needed, but in order to trigger *Mathematica* to use arbitrary-precision arithmetic. Otherwise, machine-precision arithmetic will be performed in hardware in which case *Mathematica* doesn't guarantee accuracy.

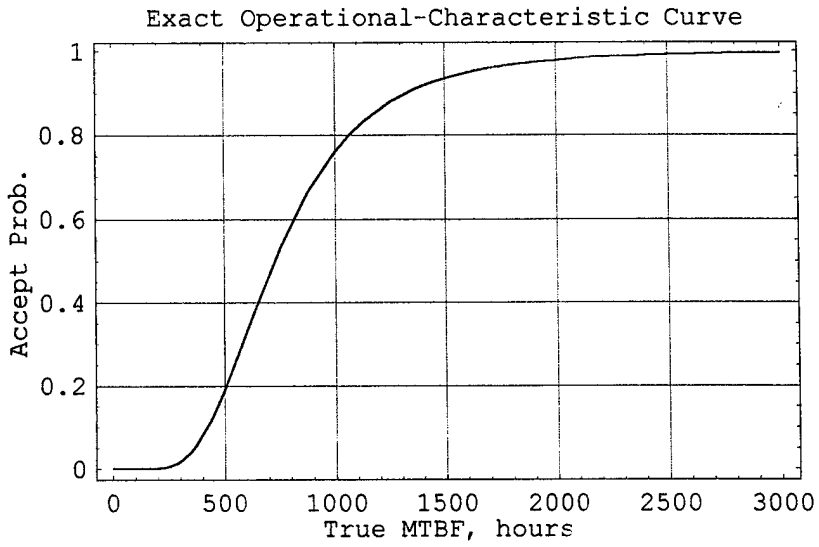
The cumulative acceptance probability for stage eight, the final stage, when *trueMTBF* is left symbolic will be generated. This is also known as the operational-characteristic function.

```
OCfunction = Sum[AcceptanceProbability[stage, trueMTBF], {stage, 1, 8}]
```

$$e^{-1395/\text{trueMTBF}} + \frac{29881094986543505 2465\, e^{-5578/\text{trueMTBF}}}{36\ \text{trueMTBF}^6} + \frac{17233046227251829\, e^{-4881/\text{trueMTBF}}}{3\ \text{trueMTBF}^5} + \frac{23887538000789\, e^{-4184/\text{trueMTBF}}}{6\ \text{trueMTBF}^4} + \frac{16613704547\, e^{-3487/\text{trueMTBF}}}{6\ \text{trueMTBF}^3} + \frac{3890655\, e^{-2789/\text{trueMTBF}}}{2\ \text{trueMTBF}^2} + \frac{1395\, e^{-2092/\text{trueMTBF}}}{\text{trueMTBF}}$$

OCfunction provides the exact acceptance probability as a function of *trueMTBF*. The exact operational-characteristic curve can now be plotted:

```
ocPlot = Plot[OCfunction, {trueMTBF, 1, 3000}, GridLines → Automatic,
   Frame → True, FrameLabel → {"True MTBF, hours", "Accept Prob.",
     "Exact Operational-Characteristic Curve" , None}];
```



It should be noted that the operational-characteristic plot is a key test-design graphic thus the plot above was assigned as the value of the symbol *ocPlot* so it could be readily inserted in Chapter 2.

- **Cumulative Acceptance Probabilities for Each Stage When *trueMTBF* Equals the Upper-Test MTBF**

It would be useful to generate a list of cumulative acceptance probabilities for all eight stages when the true MTBF equals the upper-test MTBF. The parameter *trueMTBF* will be left symbolic in order to be consistent with up-front calculations. The desired list is generated but display of the output is temporarily suppressed.

```
mycumacc = Table[
   Sum[AcceptanceProbability[stage, trueMTBF], {stage, 1, stagelim}],
   {stagelim, 1, 8}];
```

Now a table is generated which displays our calculations. A rule is used to replace *trueMTBF* with the upper-test MTBF in the cumulative acceptance probabilities stored in the list *mycumacc*:

```
NumberForm[
 TableForm[{N[timeValues, 4], N[mycumacc /. trueMTBF → 2 * 503, 4]},
  TableDirections -> {Row, Column},
  TableHeadings -> {{"Time", "Σ Accept Pr."}, Automatic},
  TableAlignments -> Center], {6, 5}]
```

	Time	Σ Accept Pr.
1.	697.	0.
2.	1395.	0.2499
3.	2092.	0.42322
4.	2789.	0.54339
5.	3487.	0.62834
6.	4184.	0.68907
7.	4881.	0.73263
8.	5578.	0.76392

In order to calculate just the final cumulative acceptance probability, we can use *OCfunction* from the previous section and employ a rule to replace *trueMTBF* with the upper-test MTBF.

```
OCfunction /. trueMTBF → 2 * 503
```

$$\frac{29881094986543505 2465}{3731559622152129 5616\, e^{2789/503}} + \frac{17233046227251829}{3091086499463328\, e^{4881/1006}} + \frac{23887538000789}{6145301191776\, e^{2092/503}} + \frac{16613704547}{6108649296\, e^{3487/1006}} + \frac{3890655}{2024072\, e^{2789/1006}} + \frac{1395}{1006\, e^{1046/503}} + \frac{1}{e^{1395/1006}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.7639210932754999180296
```

The producer risk equals one minus the acceptance probability when the true MTBF equals the upper-test MTBF. The producer risk is then one minus the answer above:

```
1 - %
```

```
0.2360789067245000819704
```

- **Cumulative Acceptance Probabilities for Each Stage When *trueMTBF* Equals the Lower-Test MTBF**

It would be useful to generate a table of cumulative acceptance probabilities for all eight stages when the true MTBF equals the lower-test MTBF. The list *mycumacc*, which was generated in the previous subsection, can be used for this purpose. A rule is used to replace *trueMTBF* with the lower-test MTBF in the cumulative acceptance probability expressions stored in the list *mycumacc*:

```
NumberForm[
 TableForm[{N[timeValues, 4], N[mycumacc /. trueMTBF → 503, 4]},
  TableDirections -> {Row, Column},
  TableHeadings -> {{"Time", "Σ Accept Pr."}, Automatic},
  TableAlignments -> Center], {6, 5}]
```

	Time	Σ Accept Pr.
1.	697.	0.
2.	1395.	0.06245
3.	2092.	0.10578
4.	2789.	0.13583
5.	3487.	0.15705
6.	4184.	0.17223
7.	4881.	0.18313
8.	5578.	0.19095

In order to calculate just the final cumulative acceptance probability, we could use the *OCfunction* from the previous section and use a rule to replace *trueMTBF* with the lower-test MTBF.

```
OCfunction /. trueMTBF → 503
```

$$\frac{29881094986543505 2465}{583056190961270244\, e^{5578/503}} + \frac{17233046227251829}{96596453108229\, e^{4881/503}} + \frac{23887538000789}{384081324486\, e^{4184/503}} + \frac{16613704547}{763581162\, e^{3487/503}} + \frac{3890655}{506018\, e^{2789/503}} + \frac{1395}{503\, e^{2092/503}} + \frac{1}{e^{1395/503}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.190952012400527518446 8
```

This is the consumer risk since the consumer risk is defined as the acceptance probability when the true MTBF equals the lower-test MTBF.

Construct Function for Continuation Probability for a Quantity of Failures

In this step, we construct a function for calculating continuation probabilities for a quantity of failures [Epstein, et al. 1963, equation 19]:

```
ContinuationProbability[stage_Integer, failure_Integer, trueMTBF_] :=
 ACProbability[stage, failure, trueMTBF]
```

Construct and Use Function for Continuation Probability for Each Stage

In this step, we construct and use a function for calculating stage-by-stage continuation probabilities [Epstein, et al. 1963, equation 21]:

```
ContinuationProbability[stage_Integer, trueMTBF_] :=
    r[stage]-1
     ∑          ContinuationProbability[stage, failure, trueMTBF] /;
failure=a[stage]+1
  a[stage] + 1 < r[stage]

ContinuationProbability[stage_Integer, trueMTBF_] :=
 0 /; Not[a[stage] + 1 < r[stage]]
```

The continuation probability for stage zero with zero failures is, by definition, one:

```
ContinuationProbability[0, trueMTBF]
```

```
1
```

- **Cumulative Continuation Probabilities for Each Stage When *trueMTBF* is Symbolic**

The cumulative continuation probability for stage one when *trueMTBF* is left symbolic is:

```
ContinuationProbability[1, trueMTBF]
```

$$e^{-697/\text{trueMTBF}} + \frac{485809\, e^{-697/\text{trueMTBF}}}{2\,\text{trueMTBF}^2} + \frac{697\, e^{-697/\text{trueMTBF}}}{\text{trueMTBF}}$$

This is an exact symbolic result. An exact result for the case where *trueMTBF* is the lower-test MTBF is:

```
% /. trueMTBF → 503
```

$$\frac{1693009}{506018\, e^{697/503}}$$

This is an exact result. An numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.8369462497111678019507
```

The cumulative continuation probability for stage eight when *trueMTBF* is symbolic is:

```
ContinuationProbability[8, trueMTBF]
```

```
0
```

This is obviously correct since the continuation probability at the last stage must be zero.

- **Cumulative Continuation Probabilities for Each Stage When *trueMTBF* Equals the Upper-Test MTBF**

The cumulative continuation probability for stage one when *trueMTBF* equals the upper-test MTBF is:

```
ContinuationProbability[1, trueMTBF] /. trueMTBF → 2 * 503
```

$$\frac{3912245}{2024072\, e^{697/1006}}$$

This is an exact result. An numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.9667233763202140309518
```

It would be useful to generate a list of cumulative continuation probabilities for all eight stages when the true MTBF equals the upper-test MTBF. The parameter *trueMTBF* will be left symbolic in order to be

consistent with up-front calculations. The desired list is generated but display of the output is temporarily suppressed.

```
mycumcon =
  Table[ContinuationProbability[stage, trueMTBF], {stage, 1, 8}];
```

Now a table is generated which displays our calculations. A rule is used to replace *trueMTBF* with the upper-test MTBF in the cumulative continuation probabilities stored in the list *mycumcon*. The cumulative acceptance probabilities for this case are also provided for reference.

```
NumberForm[
 TableForm[{N[timeValues, 4], N[mycumacc /. trueMTBF → 2 * 503, 4],
   N[mycumcon /. trueMTBF → 2 * 503, 4]},
  TableDirections → {Row, Column}, TableHeadings ->
   {{"Time", "Σ Accept Pr.", "Σ Continue Pr."}, Automatic},
  TableAlignments -> Center], {6, 5}]
```

	Time	Σ Accept Pr.	Σ Continue Pr.
1.	697.	0.	0.96672
2.	1395.	0.2499	0.68401
3.	2092.	0.42322	0.48587
4.	2789.	0.54339	0.34754
5.	3487.	0.62834	0.2494
6.	4184.	0.68907	0.15185
7.	4881.	0.73263	0.06257
8.	5578.	0.76392	0.

- **Cumulative Continuation Probabilities for Each Stage When *trueMTBF* Equals the Lower-Test MTBF**

The cumulative continuation probability for stage one when *trueMTBF* equals the lower-test MTBF is:

```
ContinuationProbability[1, trueMTBF] /. trueMTBF → 503
```

$$\frac{1693009}{506018\ e^{697/503}}$$

This is an exact result. An numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.8369462497111678019507
```

It would be useful to generate a table of cumulative continuation probabilities for all eight stages when the true MTBF equals the lower-test MTBF. The list *mycumcon*, which was generated above, can be used for this purpose. A rule is used to replace *trueMTBF* with the lower-test MTBF in the cumulative continuation probability expressions stored in the list *mycumcon*. The cumulative acceptance probabilities for this case are also provided for reference.

```
NumberForm[TableForm[{N[timeValues, 4],
   N[mycumacc /. trueMTBF → 503, 4], N[mycumcon /. trueMTBF → 503, 4]},
  TableDirections → {Row, Column}, TableHeadings ->
   {{"Time", "Σ Accept Pr.", "Σ Continue Pr."}, Automatic},
  TableAlignments -> Center], {6, 5}]
```

	Time	Σ Accept Pr.	Σ Continue Pr.
1.	697.	0.	0.83695
2.	1395.	0.06245	0.60771
3.	2092.	0.10578	0.43784
4.	2789.	0.13583	0.31508
5.	3487.	0.15705	0.22654
6.	4184.	0.17223	0.10829
7.	4881.	0.18313	0.03129
8.	5578.	0.19095	0.

Calculate Rejection Probability for Each Stage

In this step, we calculate stage-by-stage rejection probabilities using Epstein, et al. 1963, equation 22:

$$\sum_{stage=1}^{n} (ContinuationProbability[stage - 1, trueMTBF] - ContinuationProbability[stage, trueMTBF] - AcceptanceProbability[stage, trueMTBF])$$

- **Cumulative Rejection Probabilities for Each Stage When *trueMTBF* is Symbolic**

The cumulative rejection probability for stage one when *trueMTBF* is left symbolic is:

```
∑ (ContinuationProbability[stage - 1, trueMTBF] -
stage=1
    ContinuationProbability[stage, trueMTBF] -
    AcceptanceProbability[stage, trueMTBF])
```

$$1 - e^{-697/\text{trueMTBF}} - \frac{485809\, e^{-697/\text{trueMTBF}}}{2\,\text{trueMTBF}^2} - \frac{697\, e^{-697/\text{trueMTBF}}}{\text{trueMTBF}}$$

This is an exact, but partially symbolic result. An exact result can be obtained for a specific value of *trueMTBF* such as the lower-test MTBF as follows:

```
% /. trueMTBF → 503
```

$$1 - \frac{1693009}{506018\, e^{697/503}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.1630537502888321980493
```

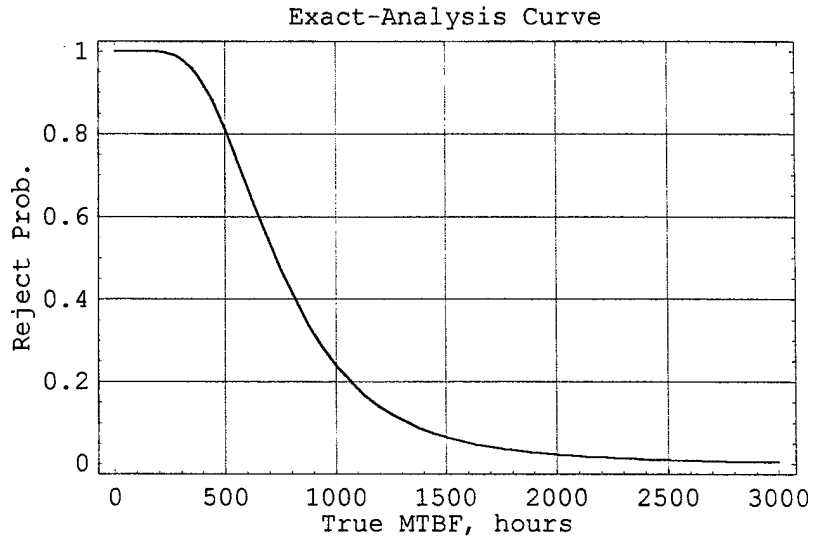
Next, the cumulative rejection probability for stage eight when *trueMTBF* is symbolic will be generated. This is one minus the operational-characteristic function.

```
rejfunction = ∑ (ContinuationProbability[stage - 1, trueMTBF] -
              stage=1
    ContinuationProbability[stage, trueMTBF] -
    AcceptanceProbability[stage, trueMTBF])
```

$$1 - e^{-1395/\text{trueMTBF}} - \frac{29881094986543505 2465\, e^{-5578/\text{trueMTBF}}}{36\,\text{trueMTBF}^6} - \frac{17233046227251829\, e^{-4881/\text{trueMTBF}}}{3\,\text{trueMTBF}^5} - \frac{23887538000789\, e^{-4184/\text{trueMTBF}}}{6\,\text{trueMTBF}^4} - \frac{16613704547\, e^{-3487/\text{trueMTBF}}}{6\,\text{trueMTBF}^3} - \frac{3890655\, e^{-2789/\text{trueMTBF}}}{2\,\text{trueMTBF}^2} - \frac{1395\, e^{-2092/\text{trueMTBF}}}{\text{trueMTBF}}$$

rejfunction provides the exact rejection probability as a function of *trueMTBF*. This function can now be plotted:

```
Plot[rejfunction, {trueMTBF, 1, 3000}, GridLines → Automatic,
  Frame → True, FrameLabel → {"True MTBF, hours",
    "Reject Prob.", "Exact-Analysis Curve" , None}];
```



- **Cumulative Rejection Probabilities for Each Stage When *trueMTBF* Equals the Upper-Test MTBF**

It would be useful to generate a list of cumulative rejection probabilities for all eight stages when the true MTBF equals the upper-test MTBF. The parameter *trueMTBF* will be left symbolic in order to be consistent with up-front calculations. The desired list is generated but display of the output is temporarily suppressed.

```
mycumrej = Table[ Sum[stage=1 to stagelim] (ContinuationProbability[stage - 1, trueMTBF] -
      ContinuationProbability[stage, trueMTBF] -
      AcceptanceProbability[stage, trueMTBF]), {stagelim, 1, 8}];
```

Now a table is generated which displays our calculations. A rule is used to replace *trueMTBF* with the upper-test MTBF in the cumulative rejection probabilities stored in the list *mycumrej*. The cumulative acceptance and continuation probabilities for this case are also provided for reference.

```
utMTBFtable = NumberForm[
  TableForm[{N[timeValues, 4], N[mycumacc /. trueMTBF → 2 * 503, 4],
    N[mycumcon /. trueMTBF → 2 * 503, 4],
    N[mycumrej /. trueMTBF → 2 * 503, 4]},
   TableDirections -> {Row, Column}, TableHeadings ->
    {{"Time", "Σ Accept Pr.", "Σ Continue Pr.", "Σ Reject Pr."},
     Automatic}, TableAlignments -> Center], {6, 5}]
```

	Time	Σ Accept Pr.	Σ Continue Pr.	Σ Reject Pr.
1.	697.	0.	0.96672	0.03328
2.	1395.	0.2499	0.68401	0.06609
3.	2092.	0.42322	0.48587	0.09091
4.	2789.	0.54339	0.34754	0.10908
5.	3487.	0.62834	0.2494	0.12227
6.	4184.	0.68907	0.15185	0.15908
7.	4881.	0.73263	0.06257	0.2048
8.	5578.	0.76392	0.	0.23608

Inspection of the table above reveals that each row sums to one as it must. The table of stage-by-stage accept, continue and reject probabilities is a key test-design graphic, thus it was assigned as the value of the symbol *utMTBFtable* so it could be readily inserted in Chapter 2.

In order to calculate just the final cumulative rejection probability, we can use *rejfunction* from the previous section and employ a rule to replace *trueMTBF* with the upper-test MTBF.

```
rejfunction /. trueMTBF → 2 * 503
```

$$1 - \frac{29881094986543505 2465}{37315596221521295616\, e^{2789/503}} - \frac{17233046227251829}{3091086499463328\, e^{4881/1006}} - \frac{23887538000789}{6145301191776\, e^{2092/503}} - \frac{16613704547}{6108649296\, e^{3487/1006}} - \frac{3890655}{2024072\, e^{2789/1006}} - \frac{1395}{1006\, e^{1046/503}} - \frac{1}{e^{1395/1006}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.2360789067245000819704
```

This is the producer risk.

- **Cumulative Rejection Probabilities for Each Stage When *trueMTBF* Equals the Lower-Test MTBF**

It would be useful to generate a table of rejection acceptance probabilities for all eight stages when the true MTBF equals the lower-test MTBF. The list *mycumrej*, which was generated in the previous subsection, can be used for this purpose. A rule is used to replace *trueMTBF* with the lower-test MTBF in the cumulative rejection probability expressions stored in the list *mycumrej*. The cumulative acceptance and continuation probabilities for this case are also provided for reference.

```
ltMTBFtable = NumberForm[
  TableForm[{N[timeValues, 4], N[mycumacc /. trueMTBF → 503, 4],
    N[mycumcon /. trueMTBF → 503, 4], N[mycumrej /. trueMTBF → 503, 4]},
   TableDirections -> {Row, Column}, TableHeadings ->
    {{"Time", "Σ Accept Pr.", "Σ Continue Pr.", "Σ Reject Pr."},
     Automatic}, TableAlignments -> Center], {6, 5}]
```

	Time	Σ Accept Pr.	Σ Continue Pr.	Σ Reject Pr.
1.	697.	0.	0.83695	0.16305
2.	1395.	0.06245	0.60771	0.32984
3.	2092.	0.10578	0.43784	0.45638
4.	2789.	0.13583	0.31508	0.5491
5.	3487.	0.15705	0.22654	0.6164
6.	4184.	0.17223	0.10829	0.71948
7.	4881.	0.18313	0.03129	0.78559
8.	5578.	0.19095	0.	0.80905

Each row sums to one as it should. The table is assigned as the value of the symbol *ltMTBFtable* so that it can be easily inserted in Chapter 2.

In order to calculate just the final cumulative rejection probability, we could use the `rejfunction` from the previous section and use a rule to replace *trueMTBF* with the lower-test MTBF.

```
rejfunction /. trueMTBF → 503
```

$$1 - \frac{29881094986543505 2465}{583056190961270244\, e^{5578/503}} - \frac{17233046227251829}{96596453108229\, e^{4881/503}} - \frac{23887538000789}{384081324486\, e^{4184/503}} - \frac{16613704547}{763581162\, e^{3487/503}} - \frac{3890655}{506018\, e^{2789/503}} - \frac{1395}{503\, e^{2092/503}} - \frac{1}{e^{1395/503}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.8090479875994724815532
```

The consumer risk equals one minus the rejection probability when the true MTBF equals the lower-test MTBF. This is one minus the answer above:

```
1 - %
```

```
0.1909520124005275184468
```

Calculate Expected Quantity of Failures and Test Time

We need to define a function for the probability that the test will terminate with an accept decision at a specified number of failures [Epstein, et al. 1963, equation 33]. First the general case and then the special case:

```
AccProbabilityF[failure_Integer, trueMTBF_] :=
 Module[{stage = 1}, While[failure > a[stage], stage++]; Which[
    failure > Last[Last[accept]], 0, 0 ≤ failure ≤ Last[Last[accept]],
    AcceptanceProbability[stage, failure, trueMTBF]]] /;
  failure ≤ a[Length[timeValues]]


AccProbabilityF[failure_Integer, trueMTBF_] :=
 0 /; failure > a[Length[timeValues]]
```

Now, we will define a function for the probability that the test will terminate with a reject decision at a specified number of failures [Epstein, et al. 1963 equation 34]:

```
RejProbabilityF[failure_Integer, trueMTBF_] :=
 Module[{rejectlist}, rejectlist =
   Select[Table[{stage, r[stage]}, {stage, 1, Length[timeValues]}],
     #[[2]] == failure &] /. {st_Integer, rej_Integer} → st;
  Which[Length[rejectlist] == 0, 0, Length[rejectlist] > 0,
   Sum[(ContinuationProbability[stage - 1, trueMTBF] -
      ContinuationProbability[stage, trueMTBF] -
      AcceptanceProbability[stage, trueMTBF]),
    {stage, First[rejectlist], Last[rejectlist]}]]]
```

The probability that the test will terminate with zero failures and a reject decision is:

```
RejProbabilityF[0, trueMTBF]
```

0

This is obviously correct since the first path to rejection is if three failures occur quickly. The probability that the test will terminate with one or two failures and a reject decision must also be zero:

```
RejProbabilityF[1, trueMTBF]
```

0

```
RejProbabilityF[2, trueMTBF]
```

0

The probability that the test will terminate with three failures and a reject decision is:

```
RejProbabilityF[3, trueMTBF]
```

$$1 - e^{-697/\text{trueMTBF}} - \frac{485809\, e^{-697/\text{trueMTBF}}}{2\,\text{trueMTBF}^2} - \frac{697\, e^{-697/\text{trueMTBF}}}{\text{trueMTBF}}$$

If *trueMTBF* is equal to the lower-test MTBF, we have:

```
% /. trueMTBF → 503
```

$$1 - \frac{1693009}{506018\, e^{697/503}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.1630537502888321980493
```

Now, we will define a function for the probability that the test will terminate with a specified number of failures [Epstein, et al. 1963 equation 35]:

```
TerminateProbability[failure_Integer, trueMTBF_] :=
 AccProbabilityF[failure, trueMTBF] + RejProbabilityF[failure, trueMTBF]
```

The probability that the test will terminate with zero failures in either acceptance or rejection is:

```
TerminateProbability[0, trueMTBF]
```

$$e^{-1395/trueMTBF}$$

The probability that the test will terminate with between zero and seven failures is:

```
Sum[TerminateProbability[failure, trueMTBF], {failure, 0, 7}]
```

```
1
```

This result is obviously correct since it's not possible for the test to continue beyond the seventh failure.

Next, we will define a function for the expected termination failure quantity [Epstein, et al. 1963 equation 36]:

```
ExpectedTerminationFailure[trueMTBF_] :=
 Sum[failure TerminateProbability[failure, trueMTBF], {failure, 0, r[Length[timeValues]]}]
```

A function for the expected termination failure quantity with *trueMTBF* left symbolic is:

expectedfailurefunction = ExpectedTerminationFailure[trueMTBF]

$$7\left(-\frac{29881094986543505246\,e^{-5578/\text{trueMTBF}}}{36\,\text{trueMTBF}^6}-\frac{17233046227251829\,e^{-4881/\text{trueMTBF}}}{3\,\text{trueMTBF}^5}+\frac{1730036199487099\,e^{-2789/\text{trueMTBF}}}{2\,\text{trueMTBF}^5}-\frac{23887538000789\,e^{-4184/\text{trueMTBF}}}{6\,\text{trueMTBF}^4}+\frac{12291172226983\,e^{-2789/\text{trueMTBF}}}{6\,\text{trueMTBF}^4}-\frac{16613704547\,e^{-3487/\text{trueMTBF}}}{6\,\text{trueMTBF}^3}+\frac{16613704547\,e^{-2789/\text{trueMTBF}}}{6\,\text{trueMTBF}^3}\right)+$$

$$6\left(\frac{29881094986543505246\,e^{-5578/\text{trueMTBF}}}{36\,\text{trueMTBF}^6}-\frac{1730036199487099\,e^{-2789/\text{trueMTBF}}}{2\,\text{trueMTBF}^5}-\frac{12291172226983\,e^{-2789/\text{trueMTBF}}}{6\,\text{trueMTBF}^4}+\frac{7093185960133\,e^{-2092/\text{trueMTBF}}}{12\,\text{trueMTBF}^4}-\frac{16613704547\,e^{-2789/\text{trueMTBF}}}{6\,\text{trueMTBF}^3}+\frac{4239172471\,e^{-2092/\text{trueMTBF}}}{3\,\text{trueMTBF}^3}-\frac{3890655\,e^{-2789/\text{trueMTBF}}}{2\,\text{trueMTBF}^2}+\frac{3890655\,e^{-2092/\text{trueMTBF}}}{2\,\text{trueMTBF}^2}\right)+$$

$$5\left(\frac{17233046227251829\,e^{-4881/\text{trueMTBF}}}{3\,\text{trueMTBF}^5}-\frac{7093185960133\,e^{-2092/\text{trueMTBF}}}{12\,\text{trueMTBF}^4}-\frac{4239172471\,e^{-2092/\text{trueMTBF}}}{3\,\text{trueMTBF}^3}+\frac{1188048001\,e^{-1395/\text{trueMTBF}}}{3\,\text{trueMTBF}^3}-\frac{3890655\,e^{-2092/\text{trueMTBF}}}{2\,\text{trueMTBF}^2}+\frac{1946025\,e^{-1395/\text{trueMTBF}}}{2\,\text{trueMTBF}^2}-\frac{1395\,e^{-2092/\text{trueMTBF}}}{\text{trueMTBF}}+\frac{1395\,e^{-1395/\text{trueMTBF}}}{\text{trueMTBF}}\right)+$$

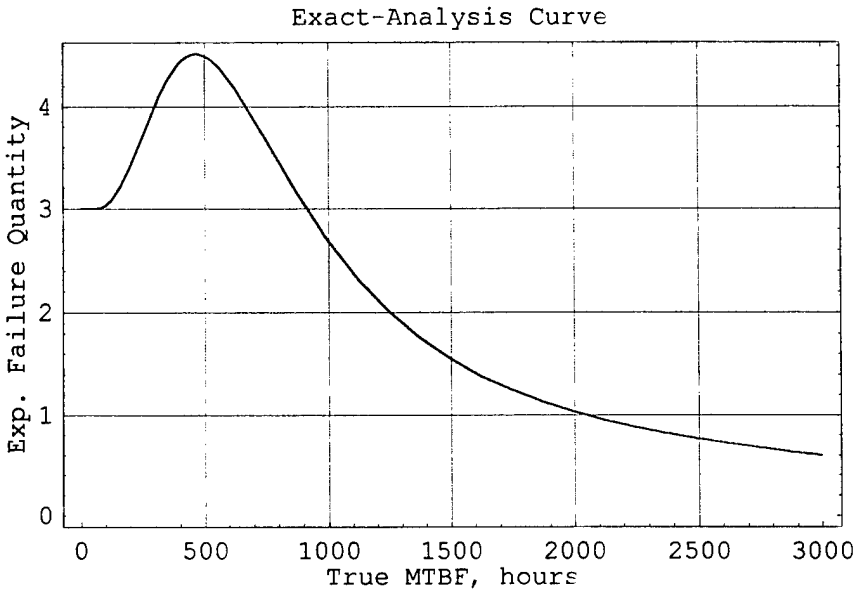
$$3\left(1-e^{-697/\text{trueMTBF}}+\frac{16613704547\,e^{-3487/\text{trueMTBF}}}{6\,\text{trueMTBF}^3}-\frac{485809\,e^{-697/\text{trueMTBF}}}{2\,\text{trueMTBF}^2}-\frac{697\,e^{-697/\text{trueMTBF}}}{\text{trueMTBF}}\right)+$$

$$4\left(-e^{-1395/\text{trueMTBF}}+e^{-697/\text{trueMTBF}}+\frac{23887538000789\,e^{-4184/\text{trueMTBF}}}{6\,\text{trueMTBF}^4}-\frac{1188048001\,e^{-1395/\text{trueMTBF}}}{3\,\text{trueMTBF}^3}-\frac{1946025\,e^{-1395/\text{trueMTBF}}}{2\,\text{trueMTBF}^2}+\frac{485809\,e^{-697/\text{trueMTBF}}}{2\,\text{trueMTBF}^2}-\frac{1395\,e^{-1395/\text{trueMTBF}}}{\text{trueMTBF}}+\frac{697\,e^{-697/\text{trueMTBF}}}{\text{trueMTBF}}\right)+$$

$$\frac{3890655\,e^{-2789/\text{trueMTBF}}}{\text{trueMTBF}^2}+\frac{1395\,e^{-2092/\text{trueMTBF}}}{\text{trueMTBF}}$$

Now we can plot this function:

```
expectedfailuresPlot = Plot[expectedfailurefunction,
   {trueMTBF, 1, 3000}, GridLines → Automatic,
   Frame → True, FrameLabel → {"True MTBF, hours",
     "Exp. Failure Quantity", "Exact-Analysis Curve" , None}];
```



The plot above, since it is a key test-design graphic, is assigned as the value of the symbol *expectedfailuresPlot* so that it can be easily inserted in Chapter 2.

In order to calculate the expected failure quantity for a true MTBF equal to the lower-test MTBF, we could use *expectedfailurefunction* and a rule to replace *trueMTBF* with this value:

```
expectedfailurefunction /. trueMTBF → 503
```

$$7\left(-\frac{298810949865435052465}{583056190961270244\, e^{5578/503}}-\frac{17233046227251829}{96596453108229\, e^{4881/503}}-\frac{23887538000789}{384081324486\, e^{4184/503}}-\frac{16613704547}{763581162\, e^{3487/503}}+\frac{15575985002365669}{193192906216458\, e^{2789/503}}\right)+6\left(\frac{298810949865435052465}{583056190961270244\, e^{5578/503}}-\frac{8530700217643112}{96596453108229\, e^{2789/503}}+\frac{7176208452385}{256054216324\, e^{2092/503}}\right)+5\left(\frac{17233046227251829}{96596453108229\, e^{4881/503}}-\frac{7886338933045}{256054216324\, e^{2092/503}}+\frac{7430333057}{763581162\, e^{1395/503}}\right)+3\left(1+\frac{16613704547}{763581162\, e^{3487/503}}-\frac{1693009}{506018\, e^{697/503}}\right)+4\left(\frac{23887538000789}{384081324486\, e^{4184/503}}-\frac{8193914219}{763581162\, e^{1395/503}}+\frac{1693009}{506018\, e^{697/503}}\right)+\frac{3890655}{253009\, e^{2789/503}}+\frac{1395}{503\, e^{2092/503}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
4.494209576902633303992
```

In order to calculate the expected failure quantity for a true MTBF equal to the upper-test MTBF, we could use *expectedfailurefunction* and a rule to replace *trueMTBF* with this value.

```
expectedfailurefunction /. trueMTBF → 2 * 503
```

$$7\left(-\frac{298810949865435052465}{37315596221521295616\, e^{2789/503}}-\frac{17233046227251829}{3091086499463328\, e^{4881/1006}}-\frac{23887538000789}{6145301191776\, e^{2092/503}}-\frac{16613704547}{6108649296\, e^{3487/1006}}+\frac{11456231651244629}{2060724332975552\, e^{2789/1006}}\right)+6\left(\frac{298810949865435052465}{37315596221521295616\, e^{2789/503}}-\frac{15417339472366109}{2060724332975552\, e^{2789/1006}}+\frac{47776513524917}{12290602383552\, e^{1046/503}}\right)+5\left(\frac{17233046227251829}{3091086499463328\, e^{4881/1006}}-\frac{64819645060757}{12290602383552\, e^{1046/503}}+\frac{4179985193}{1527162324\, e^{1395/1006}}\right)+3\left(1+\frac{16613704547}{6108649296\, e^{3487/1006}}-\frac{3912245}{2024072\, e^{697/1006}}\right)+4\left(\frac{23887538000789}{6145301191776\, e^{2092/503}}-\frac{5707147517}{1527162324\, e^{1395/1006}}+\frac{3912245}{2024072\, e^{697/1006}}\right)+\frac{3890655}{1012036\, e^{2789/1006}}+\frac{1395}{1006\, e^{1046/503}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
2.670180049154001070836
```

Next, we will define a function for the expected test time [Epstein, et al. 1963 equation 41]:

```
ExpectedTestTime[trueMTBF_] :=
 trueMTBF ExpectedTerminationFailure[trueMTBF]
```

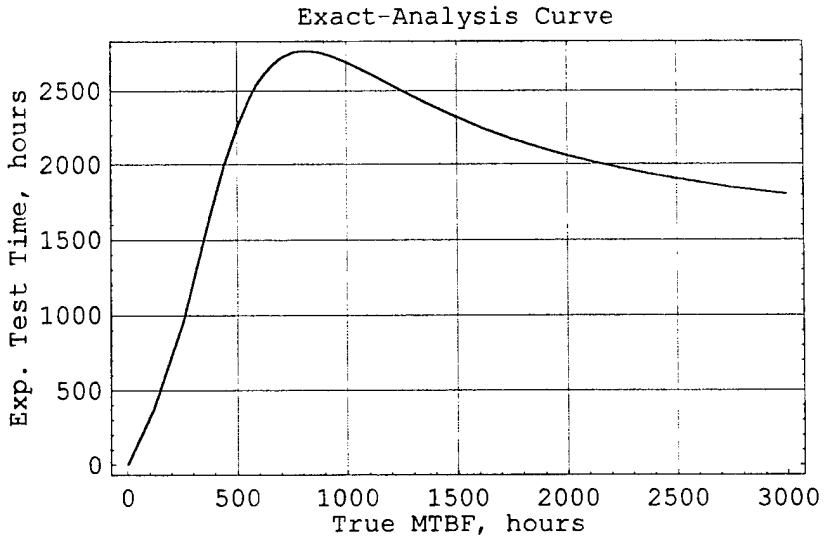
A function for the expected test time with *trueMTBF* left symbolic is:

expectedtesttimefunction = ExpectedTestTime[trueMTBF]

$$\left(7\left(-\frac{29881094986543505246 5\,e^{-5578/trueMTBF}}{36\,trueMTBF^{6}}-\frac{17233046227251829\,e^{-4881/trueMTBF}}{3\,trueMTBF^{5}}+\frac{1730036199487099\,e^{-2789/trueMTBF}}{2\,trueMTBF^{5}}-\frac{23887538000789\,e^{-4184/trueMTBF}}{6\,trueMTBF^{4}}+\frac{12291172226983\,e^{-2789/trueMTBF}}{6\,trueMTBF^{4}}-\frac{16613704547\,e^{-3487/trueMTBF}}{6\,trueMTBF^{3}}+\frac{16613704547\,e^{-2789/trueMTBF}}{6\,trueMTBF^{3}}\right)+6\left(\frac{29881094986543505246 5\,e^{-5578/trueMTBF}}{36\,trueMTBF^{6}}-\frac{1730036199487099\,e^{-2789/trueMTBF}}{2\,trueMTBF^{5}}-\frac{12291172226983\,e^{-2789/trueMTBF}}{6\,trueMTBF^{4}}+\frac{7093185960133\,e^{-2092/trueMTBF}}{12\,trueMTBF^{4}}-\frac{16613704547\,e^{-2789/trueMTBF}}{6\,trueMTBF^{3}}+\frac{4239172471\,e^{-2092/trueMTBF}}{3\,trueMTBF^{3}}-\frac{3890655\,e^{-2789/trueMTBF}}{2\,trueMTBF^{2}}+\frac{3890655\,e^{-2092/trueMTBF}}{2\,trueMTBF^{2}}\right)+5\left(\frac{17233046227251829\,e^{-4881/trueMTBF}}{3\,trueMTBF^{5}}-\frac{7093185960133\,e^{-2092/trueMTBF}}{12\,trueMTBF^{4}}-\frac{4239172471\,e^{-2092/trueMTBF}}{3\,trueMTBF^{3}}+\frac{1188048001\,e^{-1395/trueMTBF}}{3\,trueMTBF^{3}}-\frac{3890655\,e^{-2092/trueMTBF}}{2\,trueMTBF^{2}}+\frac{1946025\,e^{-1395/trueMTBF}}{2\,trueMTBF^{2}}-\frac{1395\,e^{-2092/trueMTBF}}{trueMTBF}+\frac{1395\,e^{-1395/trueMTBF}}{trueMTBF}\right)+3\left(1-e^{-697/trueMTBF}+\frac{16613704547\,e^{-3487/trueMTBF}}{6\,trueMTBF^{3}}-\frac{485809\,e^{-697/trueMTBF}}{2\,trueMTBF^{2}}-\frac{697\,e^{-697/trueMTBF}}{trueMTBF}\right)+4\left(-e^{-1395/trueMTBF}+e^{-697/trueMTBF}+\frac{23887538000789\,e^{-4184/trueMTBF}}{6\,trueMTBF^{4}}-\frac{1188048001\,e^{-1395/trueMTBF}}{3\,trueMTBF^{3}}-\frac{1946025\,e^{-1395/trueMTBF}}{2\,trueMTBF^{2}}+\frac{485809\,e^{-697/trueMTBF}}{2\,trueMTBF^{2}}-\frac{1395\,e^{-1395/trueMTBF}}{trueMTBF}+\frac{697\,e^{-697/trueMTBF}}{trueMTBF}\right)+\frac{3890655\,e^{-2789/trueMTBF}}{trueMTBF^{2}}+\frac{1395\,e^{-2092/trueMTBF}}{trueMTBF}\right)trueMTBF$$

Now we can plot this function:

```
expectedtesttimePlot = Plot[expectedtesttimefunction,
   {trueMTBF, 1, 3000}, GridLines → Automatic,
   Frame → True, FrameLabel → {"True MTBF, hours",
     "Exp. Test Time, hours", "Exact-Analysis Curve" , None}];
```



The plot above, also a key test-design graphic, is assigned as the value of the symbol *expectedtesttime-Plot* so that it can be easily inserted in Chapter 2.

In order to calculate the expected test time for a true MTBF equal to the lower-test MTBF, we could use *expectedtesttimefunction* and a rule to replace *trueMTBF* with this value.

```
expectedtesttimefunction /. trueMTBF → 503
```

$$503\left(7\left(-\frac{29881094986543505 2465}{583056190961270244\, e^{5578/503}}-\frac{17233046227251829}{96596453108229\, e^{4881/503}}-\frac{23887538000789}{384081324486\, e^{4184/503}}-\frac{16613704547}{763581162\, e^{3487/503}}+\frac{15575985002365669}{193192906216458\, e^{2789/503}}\right)+6\left(\frac{29881094986543505 2465}{583056190961270244\, e^{5578/503}}-\frac{8530700217643112}{96596453108229\, e^{2789/503}}+\frac{7176208452385}{256054216324\, e^{2092/503}}\right)+5\left(\frac{17233046227251829}{96596453108229\, e^{4881/503}}-\frac{7886338933045}{256054216324\, e^{2092/503}}+\frac{7430333057}{763581162\, e^{1395/503}}\right)+3\left(1+\frac{16613704547}{763581162\, e^{3487/503}}-\frac{1693009}{506018\, e^{697/503}}\right)+4\left(\frac{23887538000789}{384081324486\, e^{4184/503}}-\frac{8193914219}{763581162\, e^{1395/503}}+\frac{1693009}{506018\, e^{697/503}}\right)+\frac{3890655}{253009\, e^{2789/503}}+\frac{1395}{503\, e^{2092/503}}\right)$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
2260.587417182024551908
```

In order to calculate the expected test time for a true MTBF equal to the upper-test MTBF, we could use *expectedtesttimefunction* and a rule to replace *trueMTBF* with this value.

```
expectedtesttimefunction /. trueMTBF → 2 * 503
```

$$1006\left(7\left(-\frac{298810949865435052465}{37315596221521295616\,e^{2789/503}}-\frac{17233046227251829}{3091086499463328\,e^{4881/1006}}-\frac{23887538000789}{6145301191776\,e^{2092/503}}-\frac{16613704547}{6108649296\,e^{3487/1006}}+\frac{11456231651244629}{2060724332975552\,e^{2789/1006}}\right)+6\left(\frac{298810949865435052465}{37315596221521295616\,e^{2789/503}}-\frac{15417339472366109}{2060724332975552\,e^{2789/1006}}+\frac{47776513524917}{12290602383552\,e^{1046/503}}\right)+5\left(\frac{17233046227251829}{3091086499463328\,e^{4881/1006}}-\frac{64819645060757}{12290602383552\,e^{1046/503}}+\frac{4179985193}{1527162324\,e^{1395/1006}}\right)+3\left(1+\frac{16613704547}{6108649296\,e^{3487/1006}}-\frac{3912245}{2024072\,e^{697/1006}}\right)+4\left(\frac{23887538000789}{6145301191776\,e^{2092/503}}-\frac{5707147517}{1527162324\,e^{1395/1006}}+\frac{3912245}{2024072\,e^{697/1006}}\right)+\frac{3890655}{1012036\,e^{2789/1006}}+\frac{1395}{1006\,e^{1046/503}}\right)$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
2686.201129448925077261
```

Summary & Conclusions

An analysis of the exact stage-by-stage acceptance, continuation and rejection probabilities (including the impact of truncation) resulting from a sequence of reliability test plan decision rules was performed. An exact operational-characteristic function was obtained and plotted. Exact functions for expected failure quantity and test time were obtained and plotted as well.

The risks calculated in this appendix are consistent with the 20% consumer- and producer-risk goals stated by the test planners. The actual consumer and producer risks, respectively, are 19.1 and 23.6%. The exact-analysis results are consistent with the simulation documented in chapter 2.

Appendix B

Simulation Supplement to Chapter 3

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix B

Simulation Supplement to Chapter 3

Introduction

This notebook is a simulation supplement to the exact analysis contained in Chapter 3. In this notebook, the following sequence of decision rules will be simulated in order to determine their properties when θ is assumed to be equal to the lower-test value of 1,480 hours. Of particular interest is to determine whether the acceptance probability exceeds the 20% consumer-risk requirement.

The sequence of decision rules to be simulated is:

- accept at 2,382 hours if 0 failures have occurred,
- accept at 4,432 hours if 1 failure has occurred,
- accept at 6,333 hours if 2 failures have occurred,
- accept at 8,162 hours if 3 failures have occurred,
- accept at 9,947 hours if 4 failures have occurred,
- accept at 11,701 hours if 5 failures have occurred and
- reject if 6 failures occur before 11,701 hours are accumulated.

Simulation Preparation

In this section, we'll structure the decision rules so that they can be conveniently used in a simulation in the next section. First, we'll define the accept times for each potential failure quantity:

```
accept[f_ /; f == 0] = 2382
```

```
2382
```

```
accept[f_ /; f == 1] = 4432
```

```
4432
```

```
accept[f_ /; f == 2] = 6333
```

```
6333
```

```
accept[f_ /; f == 3] = 8162
```

```
8162
```

```
accept[f_ /; f == 4] = 9947
```

```
9947
```

```
accept[f_ /; f == 5] = 11701
```

```
11701
```

Next, we'll define the reject times for each potential failure quantity:

```
reject[f_ /; f == 6] = 11701
```

```
11701
```

Simulations

In this section, simulations are performed on the decision rules developed for this test plan using a true θ value of 1,480 hours. Approximate consumer risks are obtained as are approximations for the expected test time and failure quantity.

■ **Simulation Description**

Before the simulation starts, an empty list is assigned as the value of *acceptlist* and *rejectlist* thereby initializing these lists.

During the simulation, the following steps are performed for each exponential-sequential trial:

- The within-test failure counter i is initially assigned the value 1.
- The first pseudo-random failure time is generated and assigned as the initial value of *ttf*.
- While it's not the case that *ttf* meets or exceeds the accept rule for the i - 1 failure or *ttf* is less than or equal to the reject rule for the ith failure, then an additional pseudo-random failure time is added to *ttf* and i is incremented by 1.
- The While loop stops when the condition above no longer holds. Test termination has occurred.
- If the test terminated in acceptance, then the termination failure quantity is i - 1 and the termination time is the i - 1 accept time. This pair of values is appended to *acceptlist*.
- If the test terminated in rejection, then the termination failure quantity is i and the termination time is *ttf*. This pair of values is appended to *rejectlist*.

After the simulation ends, *acceptlist* contains the final failure quantity and test time of each simulated exponential-sequential test that ended in an accept decision. *rejectlist* contains the final failure quantity and test time of each simulated test that ended in a reject decision. If the true θ was assumed equal to the lower-test θ, the consumer risk is calculated as the fraction of tests that ended in acceptance. If the true θ was assumed equal to the upper-test θ, the producer risk is calculated as the fraction of tests that ended in rejection. The average quantity of failures is calculated by summing the quantity of failures that occurred during the simulation and dividing by the quantity of trials. The average test time is calculated by summing the termination times that occurred during the simulation and dividing by the quantity of trials.

The simulation function is defined next:

```
simulation[trueTheta_, trials_, prec_] :=
 Do[i = 1; ttf = -trueTheta * Log[Random[Real, {0, 1}, prec]];
While[Not[(ttf ≥ accept[i - 1]) ∨ TrueQ[ttf ≤ reject[i]]],
   ttf = ttf + -trueTheta * Log[Random[Real, {0, 1}, prec]]; i++];
If[ttf ≥ accept[i - 1], acceptlist = {acceptlist, {i - 1, accept[i - 1]}},
   Null, Null];
If[ttf ≤ reject[i], rejectlist = {rejectlist, {i, ttf}}, Null, Null],
  {trials}]
```

It should be noted that a different approach to exponential sequential simulation is taken in Chapter 2. The approach there is more efficient in terms of execution time and memory usage, thus it permits one to perform larger simulations. The approach taken here is easier to setup and saves more simulation data at the expense of additional execution time and memory.

It should also be noted that the function in the standard add-on package `Statistics`ContinuousDistributions`` for generating machine-precision, pseudorandom numbers from the exponential distribution is not used in this appendix. Instead, arbitrary-precision pseudorandom numbers are generated in order to obtain highly-accurate results as recommended by McCullough (2000). Machine-precision pseudorandom number generation is illustrated in Chapter 2.

- **First Simulation**

In this and the next three subsections, four simulations are performed assuming that the true θ equals 1,480 hours.

Start an empty list for accept decisions.

```
acceptlist = {}
```

```
{}
```

Start an empty list for reject decisions.

```
rejectlist = {}
```

```
{}
```

Run the simulation with $\lambda = \frac{1}{\theta} = \frac{1}{1480}$ for 100,000 trials using 30-digit pseudorandom number generation:

```
simulation[1480, 100000, 30]
```

Clean up the extra braces in *acceptlist*:

```
Short[acceptlist = Partition[Flatten[acceptlist], 2], 10]
```

```
<<1>>
```

Clean up the extra braces in *rejectlist*:

```
Short[rejectlist = Partition[Flatten[rejectlist], 2], 10]
```

```
{{6, 8537.69800687837419363744723035},
 {6, 5831.50636013128589753836751135},
 {6, 8739.07156713021255510916582548},
 {6, 1559.22505886098966165718481491},
 {6, 5851.15605192525127001761852342},
 <<58912>>, {6, 5068.29563539870014048753917524},
 {6, 10993.24289103998388702813974853},
 {6, 6473.40224673736399544755268559},
 {6, 6457.82157476877450541070947341}}
```

Calculate the acceptance probability. Since the true θ was assumed equal to the lower-test θ of 1480 hours, this is the consumer risk:

```
conrisk1 = Length[acceptlist] / (Length[acceptlist] + Length[rejectlist])
```

$$\frac{41079}{100000}$$

A machine-precision result is:

```
% // N
```

```
0.41079
```

Calculate the average number of failures:

```
avgfail1 = Apply[Plus, Transpose[Join[acceptlist, rejectlist]][[1]]] /
  (Length[acceptlist] + Length[rejectlist])
```

$$\frac{402349}{100000}$$

A machine-precision result is:

```
% // N
```

```
4.02349
```

Calculate the average test time:

```
avgtime1 = Apply[Plus, Transpose[Join[acceptlist, rejectlist]][[2]]] /
  (Length[acceptlist] + Length[rejectlist])
```

```
5947.24250521097294603221951379
```

- **Second Simulation**

Start an empty list for accept decisions.

```
acceptlist = {}
```

```
{}
```

Start an empty list for reject decisions.

```
rejectlist = {}
```

```
{}
```

Run the simulation with $\lambda = \frac{1}{\theta} = \frac{1}{1480}$ for 100,000 trials using 30-digit pseudorandom number generation:

```
simulation[1480, 100000, 30]
```

Clean up the extra braces in *acceptlist*:

```
Short[acceptlist = Partition[Flatten[acceptlist], 2], 10]
```

```
<<1>>
```

Clean up the extra braces in *rejectlist*:

```
Short[rejectlist = Partition[Flatten[rejectlist], 2], 10]
```

```
{{6, 5162.607196702165948578143308 74},
 {6, 7015.318130305200665701261357 19},
 {6, 7778.816983913979226050503990 99},
 {6, 10930.881760540921079785352747 30},
 {6, 4431.475852326745895672119171 86},
 <<58888>>, {6, 9027.413269388591448306749268 50},
 {6, 4820.792920017558820063617277 48},
 {6, 3265.817856205050391789324456 53},
 {6, 7177.462988322164399078667823 25}}
```

Calculate the acceptance probability. Since the true θ was assumed equal to the lower-test θ of 1480 hours, this is the consumer risk:

```
conrisk2 = Length[acceptlist] / (Length[acceptlist] + Length[rejectlist])
```

$$\frac{41103}{100000}$$

A machine-precision result is:

```
% // N
```

```
0.41103
```

Calculate the average number of failures:

```
avgfail2 = Apply[Plus, Transpose[Join[acceptlist, rejectlist]][[1]]] /
  (Length[acceptlist] + Length[rejectlist])
```

$$\frac{201261}{50000}$$

A machine-precision result is:

```
% // N
```

```
4.02522
```

Calculate the average test time:

```
avgtime2 = Apply[Plus, Transpose[Join[acceptlist, rejectlist]][[2]]] /
  (Length[acceptlist] + Length[rejectlist])
```

```
5951.346901333097168409005019 72
```

- **Third Simulation**

Start an empty list for accept decisions.

```
acceptlist = {}
```

```
{}
```

Start an empty list for reject decisions.

```
rejectlist = {}
```

```
{}
```

Run the simulation with $\lambda = \frac{1}{\theta} = \frac{1}{1480}$ for 100,000 trials using 30-digit pseudorandom number generation:

```
simulation[1480, 100000, 30]
```

Clean up the extra braces in *acceptlist*:

```
Short[acceptlist = Partition[Flatten[acceptlist], 2], 10]
```

```
{{3, 8162}, {0, 2382}, {5, 11701}, {5, 11701}, {2, 6333}, {0, 2382},
 {5, 11701}, {0, 2382}, {3, 8162}, {0, 2382}, {1, 4432}, {0, 2382},
 {4, 9947}, {0, 2382}, {0, 2382}, {5, 11701}, {0, 2382}, {1, 4432},
 {0, 2382}, {0, 2382}, {0, 2382}, {1, 4432}, {1, 4432}, {0, 2382},
 {0, 2382}, {0, 2382}, {0, 2382}, {1, 4432}, {5, 11701}, {0, 2382},
 <<41033>>, {0, 2382}, {0, 2382}, {2, 6333}, {0, 2382}, {3, 8162},
 {1, 4432}, {0, 2382}, {0, 2382}, {2, 6333}, {2, 6333}, {2, 6333},
 {2, 6333}, {1, 4432}, {3, 8162}, {0, 2382}, {0, 2382}, {1, 4432},
 {0, 2382}, {0, 2382}, {1, 4432}, {0, 2382}, {0, 2382}, {1, 4432},
 {1, 4432}, {4, 9947}, {0, 2382}, {2, 6333}, {1, 4432}, {1, 4432}}
```

Clean up the extra braces in *rejectlist*:

```
Short[rejectlist = Partition[Flatten[rejectlist], 2], 10]
```

```
{{6, 4989.93443791169813927095470199},
 {6, 9183.66085780104649793699554344},
 {6, 7123.75822296922105532263911550},
 {6, 5470.74752829537519826036407409},
 {6, 8308.09288161297634725094340221},
 <<58899>>, {6, 5624.09504810958894651445883619},
 {6, 6681.59087905510638457726550302},
 {6, 9874.16045391830694099308355778},
 {6, 11139.33039233748814435876856077}}
```

Calculate the acceptance probability. Since the true θ was assumed equal to the lower-test θ of 1480 hours, this is the consumer risk:

```
conrisk3 = Length[acceptlist] / (Length[acceptlist] + Length[rejectlist])
```

$$\frac{10273}{25000}$$

A machine-precision result is:

```
% // N
```

```
0.41092
```

Calculate the average number of failures:

```
avgfail3 = Apply[Plus, Transpose[Join[acceptlist, rejectlist]][[1]]] /
  (Length[acceptlist] + Length[rejectlist])
```

$$\frac{403253}{100000}$$

A machine-precision result is:

```
% // N
```

```
4.03253
```

Calculate the average test time:

```
avgtime3 = Apply[Plus, Transpose[Join[acceptlist, rejectlist]][[2]]] /
  (Length[acceptlist] + Length[rejectlist])
```

```
5966.7450356448848629985826826
```

- **Fourth Simulation**

Start an empty list for accept decisions.

```
acceptlist = {}
```

```
{}
```

Start an empty list for reject decisions.

```
rejectlist = {}
```

```
{}
```

Run the simulation with $\lambda = \frac{1}{\theta} = \frac{1}{1480}$ for 100,000 trials using 30-digit pseudorandom number generation:

```
simulation[1480, 100000, 30]
```

Clean up the extra braces in *acceptlist*:

```
Short[acceptlist = Partition[Flatten[acceptlist], 2], 10]
```

```
<<1>>
```

Clean up the extra braces in *rejectlist*:

```
Short[rejectlist = Partition[Flatten[rejectlist], 2], 10]
```

```
{{6, 8926.08352364327643191597133723},
 {6, 9523.05054518911528036632730384},
 {6, 4934.00253410276477355948947100},
 {6, 5257.43660580122580093609253988},
 {6, 7821.80050896919997891679184687},
 <<58642>>, {6, 6491.44002345079453726472144300},
 {6, 6577.75308534811470059153659672},
 {6, 6468.04634771090018997591283277},
 {6, 9335.30224483235148240318828980}}
```

Calculate the acceptance probability. Since the true θ was assumed equal to the lower-test θ of 1480 hours, this is the consumer risk:

```
conrisk4 = Length[acceptlist] / (Length[acceptlist] + Length[rejectlist])
```

$$\frac{41349}{100000}$$

A machine-precision result is:

```
% // N
```

```
0.41349
```

Calculate the average number of failures:

```
avgfail4 = Apply[Plus, Transpose[Join[acceptlist, rejectlist]][[1]]] /
  (Length[acceptlist] + Length[rejectlist])
```

$$\frac{50133}{12500}$$

A machine-precision result is:

```
% // N
```

```
4.01064
```

Calculate the average test time:

```
avgtime4 = Apply[Plus, Transpose[Join[acceptlist, rejectlist]][[2]]] /
  (Length[acceptlist] + Length[rejectlist])
```

```
5936.234141139068579202948351 31
```

- **Combined Results from True θ Equals 1,480 Hours Simulations**

It appears that the consumer risk is approximately between

```
Min[{conrisk1, conrisk2, conrisk3, conrisk4}] // N
```

```
0.41079
```

and

```
Max[{conrisk1, conrisk2, conrisk3, conrisk4}] // N
```

```
0.41349
```

The exact answer of 41.14% from Chapter 3 falls within this interval. The average consumer risk for all 400,000 trials is:

```
(conrisk1 + conrisk2 + conrisk3 + conrisk4) / 4 // N
```

```
0.411558
```

It appears that the expected quantity of failures is approximately between

```
Min[{avgfail1, avgfail2, avgfail3, avgfail4}] // N
```

```
4.01064
```

and

```
Max[{avgfail1, avgfail2, avgfail3, avgfail4}] // N
```

```
4.03253
```

The exact answer from Chapter 3 of 4.025 falls within this interval. The average quantity of failures for all 400,000 trials is:

$$\frac{avgfail1 + avgfail2 + avgfail3 + avgfail4}{4} // N$$

```
4.02297
```

It appears that the expected test time is approximately between

```
Min[{avgtime1, avgtime2, avgtime3, avgtime4}] // N
```

```
5936.23
```

and

```
Max[{avgtime1, avgtime2, avgtime3, avgtime4}] // N
```

```
5966.75
```

The exact answer from Chapter 3 of 5,956 hours falls within this interval. The average test time for all 400,000 trials is:

$$\frac{avgtime1 + avgtime2 + avgtime3 + avgtime4}{4}$$

```
5950.3921458320058891606888827
```

Summary

When the true θ equaled the lower-test value of 1,480 hours, the following results were obtained from 400,000 simulation trials:

- acceptance probability (i.e., consumer risk) is approximately between 41.1 and 41.3% (compared with 41.14% obtained from the exact analysis),
- expected quantity of failures is approximately between 4.01 and 4.03 (compared with 4.025 obtained from the exact analysis),
- expected quantity of test time is approximately between 5,936 and 5,967 hours (compared with 5,956 obtained from the exact analysis).

The results from these simulations are entirely consistent with, and thus constitute a rough double-check of, the exact analysis in Chapter 3.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix C

Exact Analysis of Exponential Sequential Test Plan Designed in Chapter 4

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix C

Exact Analysis of Exponential Sequential Test Plan Designed in Chapter 4

This notebook contains an analysis of the exact stage-by-stage acceptance, continuation and rejection probabilities resulting from the exponential sequential test designed in Chapter 4. Included are the important special cases that arise at the last stage: consumer risk, producer risk and the operational-characteristic curve. *Mathematica* symbolics are used to obtain results with the Mean Time Between Failures (MTBF) parameter held symbolic until a numerical value is supplied. The stage-by-stage calculations are performed in such a way that numerical errors that would otherwise accumulate are entirely avoided. The results of all calculations are "exact" but include occurrences of the exponential function. Numerical approximations to any desired precision are provided as well.

The exact-analysis method implemented in this notebook was developed by Epstein, Patterson and Qualls [1963].

Setup

The code used in this appendix is contained in the standard add-on packages *MultipleListPlot* and *DiscreteDistributions.m*, which are loaded thus:

```
Needs["Graphics`MultipleListPlot`"]
```

```
Needs["Statistics`DiscreteDistributions`"]
```

Extract Reliability Test Plan Decision Rules

In order to apply the exact-analysis method, we need to construct a list of accept points from the decision rules designed in Chapter 4. Each pair will be of the form $\{t_i, i\}$ where the first pair defines the zero-failure accept time, the second pair defines the one-failure accept time, etc. The accept rules are assigned as the value of the symbol *rawaccept*.

```
N[rawaccept = {{6100, 0}, {9150, 1}, {12199, 2},
   {15249, 3}, {18299, 4}, {20828, 5}, {20828, 6}}]

{{6100., 0.}, {9150., 1.}, {12199., 2.},
 {15249., 3.}, {18299., 4.}, {20828., 5.}, {20828., 6.}}
```

There are two occurrences of the last accept time. Since we can only use unique times, one repeat is eliminated and the pair with the highest quantity of failures is retained. The result is assigned as the value of the symbol *accept*.

```
N[accept = Delete[rawaccept, {{6}}]]

{{6100., 0.}, {9150., 1.}, {12199., 2.},
 {15249., 3.}, {18299., 4.}, {20828., 6.}}
```

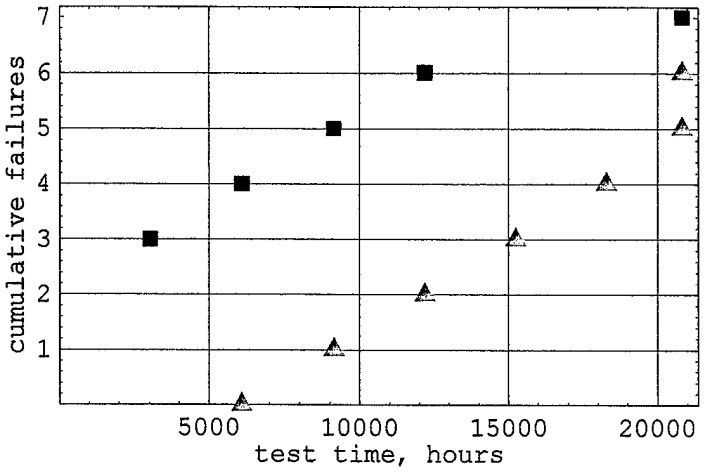
We need to construct a list of reject points from these decision rules. Each pair will be of the form $\{t_i, i\}$ where the first pair defines the shortest reject time and the corresponding quantity of failures, the second defines the second-shortest reject time and the corresponding quantity of failures, etc.

```
N[reject = {{3050, 3}, {6100, 4}, {9150, 5}, {12199, 6}, {20828, 7}}]

{{3050., 3.}, {6100., 4.}, {9150., 5.}, {12199., 6.}, {20828., 7.}}
```

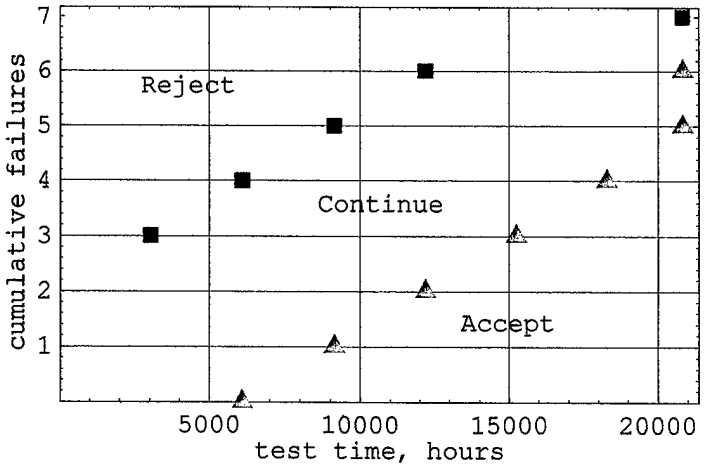
The decision rules, with the accept and reject points represented by triangles and boxes, respectively, are plotted as follows:

```
decisionPlot =
  MultipleListPlot[rawaccept, Reverse[reject], PlotJoined → False,
   PlotRange → {{0, Automatic}, {0, Automatic}}, Frame → True,
   FrameLabel → {"test time, hours", "cumulative failures"},
   GridLines → Automatic,
   SymbolShape → {PlotSymbol[Triangle, 5], PlotSymbol[Box, 3]},
   SymbolStyle → {RGBColor[0, 1, 0], RGBColor[1, 0, 0]}];
```



Identification of the reject, continue and accept regions can be overlaid thus:

```
Show[%, Graphics[{Text["Reject", Scaled[{0.2, 0.8}]], Text["Continue",
    Scaled[{0.5, 0.5}]], Text["Accept", Scaled[{0.7, 0.2}]]}]];
```



Define Function for Stage Times

In this step, we will construct a stage-time function. First, a list is needed of the times for each stage. The stage times are comprised of the accept and reject times joined into a single list and sorted from shortest to longest. The list of stage times is constructed as follows:

```
timeValues =
 Sort[Union[First[Transpose[accept]], First[Transpose[reject]]], Less]
```

```
{3050, 6100, 9150, 12199, 15249, 18299, 20828}
```

It should be noted that the times are expressed as exact numbers (i.e., either as integers or rational numbers) in order to avoid approximations until after the stage-by-stage calculations are complete. If the times are expressed in decimal form, *Mathematica* will treat them as approximate and will use machine-precision (unless many zeroes are used) arithmetic.

It should also be noted that the function `Union` was used to eliminate any repeats occurring as the two lists were combined.

The quantity of stages is:

```
Length[timeValues]
```

```
7
```

A function which will provide time values as a function of stage, except for the special case of stage zero, is:

```
t[stage_Integer /; stage > 0] := timeValues[[stage]]
```

The initial condition for time [Epstein, et al. 1963, equation 16]:

```
t[i_ /; i == 0] := 0
```

Construct Accept-Number Function

In this step, we will construct an accept-number function. First, we will generate an `InterpolatingFunction` object from *accept*:

```
fA = Interpolation[accept, InterpolationOrder → 1]

InterpolatingFunction[{{6100, 20828}}, <>]
```

Now, we define a function which will provide an integer-valued accept number for each stage using Epstein, et al. 1963, equation 11:

```
a[stage_Integer /; stage > 0] := -1 /; t[stage] < First[First[accept]]

a[stage_Integer /; stage > 0] := Floor[fA[t[stage]]]
```

A special case of the accept-number function is defined for the initial condition at stage zero [Epstein, et al. 1963, equation 16]:

```
a[stage_Integer /; stage == 0] := -1
```

Construct Reject-Number Function

In this step, we will construct a reject-number function. First, we will generate an `InterpolatingFunction` object from *reject*:

```
fR = Interpolation[reject, InterpolationOrder → 1]

InterpolatingFunction[{{3050, 20828}}, <>]
```

Now, we define an function which will provide an integer-valued reject number for each stage using Epstein, et al. 1963, equation 12:

```
r[stage_Integer /; stage > 0] := Ceiling[fR[t[stage]]]
```

A special case of the reject-number function is defined for the initial condition at stage zero:

```
r[stage_ /; stage == 0] := 1
```

Tabulation of Accept, Continuation and Reject Points

In this step, we generate a table of accept, continuation and reject numbers. This is done to provide a convenient stage-by-stage listing of the test plan to be analyzed. The table is generated as follows:

```
TableForm[Transpose[{Range[Length[timeValues]],
   Table[N[t[stage]], {stage, 1, Length[timeValues]}],
   Table[a[stage], {stage, 1, Length[timeValues]}],
   Append[Table[a[stage] + 1, {stage, 1, Length[timeValues] - 1}], NA],
   Append[Table[r[stage] - 1, {stage, 1, Length[timeValues] - 1}], NA],
   Table[r[stage], {stage, 1, Length[timeValues]}]}],
 TableHeadings -> {None, {"Stage", "Time", "Accept",
    "Continue (min)", "Continue (max)", "Reject"}},
 TableSpacing → {1, 1.5}, TableAlignments → Center]
```

Stage	Time	Accept	Continue (min)	Continue (max)	Reject
1	3050.	-1	0	2	3
2	6100.	0	1	3	4
3	9150.	1	2	4	5
4	12199.	2	3	5	6
5	15249.	3	4	6	7
6	18299.	4	5	6	7
7	20828.	6	NA	NA	7

Construct Function for Acceptance/Continuation Probability for a Quantity of Failures

In this step, we construct a function for calculating acceptance/continuation probabilities for a quantity of failures [Epstein, et al. 1963, equation 17]:

```
ACProbability[stage_, failure_, trueMTBF_] /;
  And[stage > 0, (a[stage - 1] + 1) ≤ failure ≤ r[stage] - 1] :=
 aclist[stage, failure, trueMTBF]
```

```
ACProbability[stage_, failure_, trueMTBF_] /;
  And[stage > 0, Not[(a[stage - 1] + 1) ≤ failure ≤ r[stage] - 1]] := 0
```

Two initial conditions for this function are also needed [Epstein, et al. 1963, equation 16]:

```
ACProbability[0, 0, trueMTBF_] := 1
```

```
ACProbability[0, failure_Integer /; failure > 0, trueMTBF_] := 0
```

Up-front Calculation of Acceptance/Continuation Probabilities

In order to reduce execution time, stage-by-stage calculations of acceptance and continuation probabilities are developed in this step.

The calculation of acceptance/continuation probabilities at stage *i* use the results from stage *i* -1 [Epstein, et al. 1963, equation 17]. The most natural approach with *Mathematica* may be to proceed from the last stage to the first using recursion. This approach results in execution times that grow exponentially with the number of stages required by the test plan under analysis. Acceptance/continuation probabilities calculated at one stage are needed many times at later stages. Consequently, it was necessary to calculate and store the acceptance/continuation probabilities for each stage in succession from stages 1 through *n*.

A function for building up the calculations is:

```
aclistfunction[stage_Integer, failure_Integer, trueMTBF_] :=
 aclist[stage, failure, trueMTBF] =
    Sum[ACProbability[stage - 1, j, trueMTBF]
      PDF[PoissonDistribution[(t[stage] - t[stage - 1])/trueMTBF],
       failure - j], {j, a[stage-1]+1, failure}]
```

An indexed variable *aclist* is used to build up the acceptance and continuation probabilities.

The acceptance and continuation points for the stages are:

```
Map[aclistfunction[1, #, trueMTBF] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 1]]
```

$$\left\{e^{-3050/\text{trueMTBF}},\ \frac{3050\ e^{-3050/\text{trueMTBF}}}{\text{trueMTBF}},\ \frac{4651250\ e^{-3050/\text{trueMTBF}}}{\text{trueMTBF}^2}\right\}$$

```
Map[aclistfunction[2, #, trueMTBF] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 2]]
```

$$\left\{e^{-6100/\text{trueMTBF}},\ \frac{6100\ e^{-6100/\text{trueMTBF}}}{\text{trueMTBF}},\ \frac{18605000\ e^{-6100/\text{trueMTBF}}}{\text{trueMTBF}^2},\ \frac{99304187500\ e^{-6100/\text{trueMTBF}}}{3\ \text{trueMTBF}^3}\right\}$$

```
Map[aclistfunction[3, #, trueMTBF] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 3]]
```

$$\left\{\frac{6100\, e^{-9150/\text{trueMTBF}}}{\text{trueMTBF}},\ \frac{37210000\, e^{-9150/\text{trueMTBF}}}{\text{trueMTBF}^2},\right.$$
$$\left.\frac{354657812500\, e^{-9150/\text{trueMTBF}}}{3\ \text{trueMTBF}^3},\ \frac{216341265625000\, e^{-9150/\text{trueMTBF}}}{\text{trueMTBF}^4}\right\}$$

```
Map[aclistfunction[4, #, trueMTBF] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 4]]
```

$$\left\{\frac{37210000\, e^{-12199/\text{trueMTBF}}}{\text{trueMTBF}^2},\ \frac{695017682500\, e^{-12199/\text{trueMTBF}}}{3\ \text{trueMTBF}^3},\right.$$
$$\left.\frac{2249254089002500\, e^{-12199/\text{trueMTBF}}}{3\ \text{trueMTBF}^4},\ \frac{4154747817367926250\, e^{-12199/\text{trueMTBF}}}{3\ \text{trueMTBF}^5}\right\}$$

```
Map[aclistfunction[5, #, trueMTBF] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 5]]
```

$$\left\{\frac{695017682500\, e^{-15249/\text{trueMTBF}}}{3\ \text{trueMTBF}^3},\ \frac{1456352673542500\, e^{-15249/\text{trueMTBF}}}{\text{trueMTBF}^4},\right.$$
$$\frac{14247673784553676250\, e^{-15249/\text{trueMTBF}}}{3\ \text{trueMTBF}^5},$$
$$\left.\frac{7926120981030594081250\, e^{-15249/\text{trueMTBF}}}{9\ \text{trueMTBF}^6}\right\}$$

```
Map[aclistfunction[6, #, trueMTBF] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 6]]
```

$$\left\{\frac{1456352673542500\, e^{-18299/\text{trueMTBF}}}{\text{trueMTBF}^4},\ \frac{27573300747467551250\, e^{-18299/\text{trueMTBF}}}{3\ \text{trueMTBF}^5},\right.$$
$$\left.\frac{2705921682943030566250000\, e^{-18299/\text{trueMTBF}}}{9\ \text{trueMTBF}^6}\right\}$$

```
Map[aclistfunction[7, #, trueMTBF] &,
 Apply[Range, {a[i - 1] + 1, r[i] - 1} /. i → 7]]
```

$$\left\{\frac{27573300747467551250\, e^{-20828/\text{trueMTBF}}}{3\ \text{trueMTBF}^5},\right.$$
$$\left.\frac{4797908010653393679587500\, e^{-20828/\text{trueMTBF}}}{9\ \text{trueMTBF}^6}\right\}$$

Construct Function for Acceptance Probability for a Quantity of Failures

In this step, we construct a function for calculating acceptance probabilities for a quantity of failures [Epstein, et al. 1963, equation 18]:

```
AcceptanceProbability[stage_Integer, failure_Integer, trueMTBF_] :=
 ACProbability[stage, failure, trueMTBF]
```

Construct and Use Function for Acceptance Probability for Each Stage

In this step, we construct and use a function for calculating stage-by-stage acceptance probabilities [Epstein, et al. 1963, equation 20]:

```
AcceptanceProbability[stage_Integer, trueMTBF_] :=
     a[stage]
      ∑          AcceptanceProbability[stage, failure, trueMTBF] /;
failure=a[stage-1]+1
  a[stage - 1] < a[stage]

AcceptanceProbability[stage_Integer, trueMTBF_] :=
 0 /; Not[a[stage - 1] < a[stage]]
```

The acceptance probability as a function of true MTBF is the sum of the probabilities of acceptance at each stage. This is given by Epstein, et al. 1963, equation 14:

```
AcceptanceProbability[trueMTBF_] :=
  n
  ∑ AcceptanceProbability[trueMTBF_, i_]
 i=1
```

- **Cumulative Acceptance Probabilities for Each Stage When *trueMTBF* is Symbolic**

The cumulative acceptance probability for stage one when *trueMTBF* is left symbolic is:

```
  1
  ∑  AcceptanceProbability[stage, trueMTBF]
stage=1
```

```
0
```

This result is correct since the first opportunity for acceptance to occur is at stage two. The cumulative acceptance probability for stage two when *trueMTBF* is left symbolic is:

```
∑ AcceptanceProbability[stage, trueMTBF]
stage=1
```

$$e^{-6100/\text{trueMTBF}}$$

The result above is exact but partially symbolic. An exact result can be obtained for a specific value of *trueMTBF* such as the lower-test MTBF as follows:

```
% /. trueMTBF → 2200
```

$$\frac{1}{e^{61/22}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.062491341194386502695 95
```

An answer accurate to 22 decimal places was elicited not because an answer this precise was needed, but in order to trigger *Mathematica* to use arbitrary-precision arithmetic. Otherwise, machine-precision arithmetic will be performed in hardware in which case *Mathematica* doesn't guarantee accuracy.

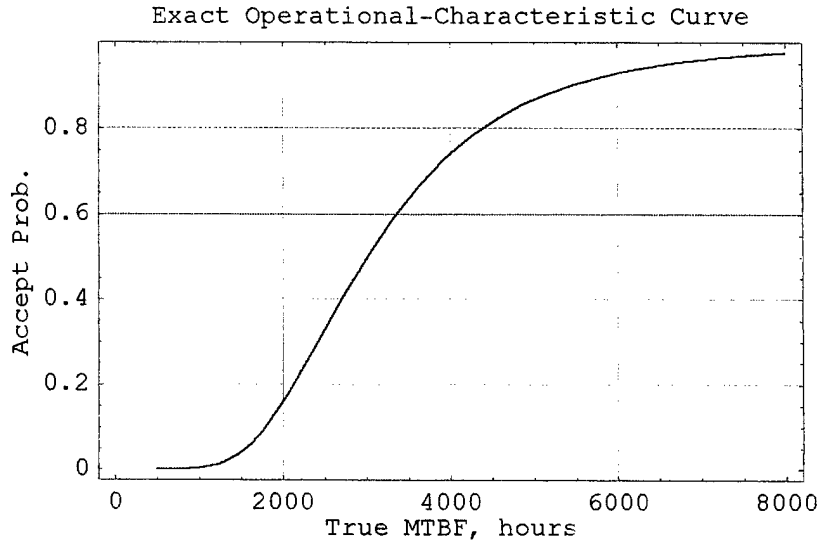
The cumulative acceptance probability for stage seven, the final stage, when *trueMTBF* is left symbolic will be generated. This is also known as the operational-characteristic function.

```
OCfunction = ∑ AcceptanceProbability[stage, trueMTBF]
             stage=1
```

$$e^{-6100/\text{trueMTBF}} + \frac{47979080106533936795 8750\, e^{-20828/\text{trueMTBF}}}{9\ \text{trueMTBF}^6} + \frac{2757300747467551250\, e^{-20828/\text{trueMTBF}}}{3\ \text{trueMTBF}^5} + \frac{1456352673542500\, e^{-18299/\text{trueMTBF}}}{\text{trueMTBF}^4} + \frac{695017682500\, e^{-15249/\text{trueMTBF}}}{3\ \text{trueMTBF}^3} + \frac{37210000\, e^{-12199/\text{trueMTBF}}}{\text{trueMTBF}^2} + \frac{6100\, e^{-9150/\text{trueMTBF}}}{\text{trueMTBF}}$$

OCfunction provides the exact acceptance probability as a function of *trueMTBF*. The exact operational-characteristic curve can now be plotted:

```
ocPlot = Plot[OCfunction, {trueMTBF, 500, 8000}, GridLines → Automatic,
   Frame → True, FrameLabel → {"True MTBF, hours", "Accept Prob.",
     "Exact Operational-Characteristic Curve" , None}];
```



It should be noted that the operational-characteristic plot is a key test-design graphic thus the plot above was assigned as the value of the symbol *ocPlot* so it could be readily inserted in Chapter 4.

- **Cumulative Acceptance Probabilities for Each Stage When *trueMTBF* Equals the Upper-Test MTBF**

It would be useful to generate a list of cumulative acceptance probabilities for all seven stages when the true MTBF equals the upper-test MTBF. The parameter *trueMTBF* will be left symbolic in order to be consistent with up-front calculations. The desired list is generated but display of the output is temporarily suppressed.

```
mycumacc = Table[
   Sum[AcceptanceProbability[stage, trueMTBF], {stage, 1, stagelim}], {stagelim, 1, 7}];
```

Now a table is generated which displays our calculations. A rule is used to replace *trueMTBF* with the upper-test MTBF in the cumulative acceptance probabilities stored in the list *mycumacc*:

```
NumberForm[
 TableForm[{N[timeValues, 4], N[mycumacc /. trueMTBF → 4400, 4]},
  TableDirections -> {Row, Column},
  TableHeadings -> {{"Time", "Σ Accept Pr."}, Automatic},
  TableAlignments -> Center], {6, 5}]
```

	Time	Σ Accept Pr.
1.	3050.	0.
2.	6100.	0.24998
3.	9150.	0.42326
4.	12199.	0.5434
5.	15249.	0.62839
6.	18299.	0.6891
7.	20828.	0.80273

In order to calculate just the final cumulative acceptance probability, we can use *OCfunction* from the previous section and employ a rule to replace *trueMTBF* with the upper-test MTBF.

```
OCfunction /. trueMTBF → 4400
```

$$\frac{61364245158684439597}{4749587251200000000\, e^{5207/1100}} + \frac{582541069417}{149923840000\, e^{18299/4400}} + \frac{278007073}{102220800\, e^{15249/4400}} + \frac{3721}{1936\, e^{1109/400}} + \frac{61}{44\, e^{183/88}} + \frac{1}{e^{61/44}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.8027270619907664067051
```

The producer risk equals one minus the acceptance probability when the true MTBF equals the upper-test MTBF. The producer risk is then one minus the answer above:

```
1 - %
```

```
0.19727293800923359329 49
```

- **Cumulative Acceptance Probabilities for Each Stage When *trueMTBF* Equals the Lower-Test MTBF**

It would be useful to generate a table of cumulative acceptance probabilities for all seven stages when the true MTBF equals the lower-test MTBF. The list *mycumacc*, which was generated in the previous subsection, can be used for this purpose. A rule is used to replace *trueMTBF* with the lower-test MTBF in the cumulative acceptance probability expressions stored in the list *mycumacc*:

```
NumberForm[
 TableForm[{N[timeValues, 4], N[mycumacc /. trueMTBF → 2200, 4]},
  TableDirections -> {Row, Column},
  TableHeadings -> {{"Time", "Σ Accept Pr."}, Automatic},
  TableAlignments -> Center], {6, 5}]
```

	Time	Σ Accept Pr.
1.	3050.	0.
2.	6100.	0.06249
3.	9150.	0.10581
4.	12199.	0.13584
5.	15249.	0.15709
6.	18299.	0.17227
7.	20828.	0.22243

In order to calculate just the final cumulative acceptance probability, we could use the *OCfunction* from the previous section and use a rule to replace *trueMTBF* with the lower-test MTBF.

```
OCfunction /. trueMTBF → 2200
```

$$\frac{48129060799900014997}{74212300800000000\ e^{5207/550}} + \frac{582541069417}{9370240000\ e^{18299/2200}} + \frac{278007073}{12777600\ e^{15249/2200}} + \frac{3721}{484\ e^{1109/200}} + \frac{61}{22\ e^{183/44}} + \frac{1}{e^{61/22}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.2224300264826413065622
```

This is the consumer risk since the consumer risk is defined as the acceptance probability when the true MTBF equals the lower-test MTBF.

Construct Function for Continuation Probability for a Quantity of Failures

In this step, we construct a function for calculating continuation probabilities for a quantity of failures [Epstein, et al. 1963, equation 19]:

```
ContinuationProbability[stage_Integer, failure_Integer, trueMTBF_] :=
 ACProbability[stage, failure, trueMTBF]
```

Construct and Use Function for Continuation Probability for Each Stage

In this step, we construct and use a function for calculating stage-by-stage continuation probabilities [Epstein, et al. 1963, equation 21]:

```
ContinuationProbability[stage_Integer, trueMTBF_] :=
    r[stage]-1
     ∑          ContinuationProbability[stage, failure, trueMTBF] /;
failure=a[stage]+1
  a[stage] + 1 < r[stage]

ContinuationProbability[stage_Integer, trueMTBF_] :=
 0 /; Not[a[stage] + 1 < r[stage]]
```

The continuation probability for stage zero with zero failures is, by definition, one:

```
ContinuationProbability[0, trueMTBF]
```

```
1
```

- **Cumulative Continuation Probabilities for Each Stage When *trueMTBF* is Symbolic**

The cumulative continuation probability for stage one when *trueMTBF* is left symbolic is:

```
ContinuationProbability[1, trueMTBF]
```

$$e^{-3050/\text{trueMTBF}} + \frac{4651250\, e^{-3050/\text{trueMTBF}}}{\text{trueMTBF}^2} + \frac{3050\, e^{-3050/\text{trueMTBF}}}{\text{trueMTBF}}$$

This is an exact symbolic result. An exact result for the case where *trueMTBF* is the lower-test MTBF is:

```
% /. trueMTBF → 2200
```

$$\frac{12961}{3872\, e^{61/44}}$$

This is an exact result. An numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.8367834552340778454066
```

The cumulative continuation probability for stage seven when *trueMTBF* is symbolic is:

```
ContinuationProbability[7, trueMTBF]
```

```
0
```

This is obviously correct since the continuation probability at the last stage must be zero.

- **Cumulative Continuation Probabilities for Each Stage When *trueMTBF* Equals the Upper-Test MTBF**

The cumulative continuation probability for stage one when *trueMTBF* equals the upper-test MTBF is:

```
ContinuationProbability[1, trueMTBF] /. trueMTBF → 4400
```

$$\frac{29945}{15488\, e^{61/88}}$$

This is an exact result. An numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.9666826831862179957211
```

It would be useful to generate a list of cumulative continuation probabilities for all seven stages when the true MTBF equals the upper-test MTBF. The parameter *trueMTBF* will be left symbolic in order to be consistent with up-front calculations. The desired list is generated but display of the output is temporarily suppressed.

```
mycumcon =
  Table[ContinuationProbability[stage, trueMTBF], {stage, 1, 7}];
```

Now a table is generated which displays our calculations. A rule is used to replace *trueMTBF* with the upper-test MTBF in the cumulative continuation probabilities stored in the list *mycumcon*. The cumulative acceptance probabilities for this case are also provided for reference.

```
NumberForm[TableForm[{N[timeValues, 4],
   N[mycumacc /. trueMTBF → 4400, 4], N[mycumcon /. trueMTBF → 4400, 4]},
  TableDirections → {Row, Column}, TableHeadings ->
   {{"Time", "Σ Accept Pr.", "Σ Continue Pr."}, Automatic},
  TableAlignments -> Center], {6, 5}]
```

	Time	Σ Accept Pr.	Σ Continue Pr.
1.	3050.	0.	0.96668
2.	6100.	0.24998	0.68394
3.	9150.	0.42326	0.48583
4.	12199.	0.5434	0.34752
5.	15249.	0.62839	0.24936
6.	18299.	0.6891	0.15182
7.	20828.	0.80273	0.

- **Cumulative Continuation Probabilities for Each Stage When *trueMTBF* Equals the Lower-Test MTBF**

The cumulative continuation probability for stage one when *trueMTBF* equals the lower-test MTBF is:

```
ContinuationProbability[1, trueMTBF] /. trueMTBF → 2200
```

$$\frac{12961}{3872\, e^{61/44}}$$

This is an exact result. An numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.8367834552340778454066
```

It would be useful to generate a table of cumulative continuation probabilities for all seven stages when the true MTBF equals the lower-test MTBF. The list *mycumcon*, which was generated above, can be

used for this purpose. A rule is used to replace *trueMTBF* with the lower-test MTBF in the cumulative continuation probability expressions stored in the list *mycumcon*. The cumulative acceptance probabilities for this case are also provided for reference.

```
NumberForm[TableForm[{N[timeValues, 4],
   N[mycumacc /. trueMTBF → 2200, 4], N[mycumcon /. trueMTBF → 2200, 4]},
  TableDirections → {Row, Column}, TableHeadings ->
   {{"Time", "Σ Accept Pr.", "Σ Continue Pr."}, Automatic},
  TableAlignments -> Center], {6, 5}]
```

	Time	Σ Accept Pr.	Σ Continue Pr.
1.	3050.	0.	0.83678
2.	6100.	0.06249	0.60776
3.	9150.	0.10581	0.43781
4.	12199.	0.13584	0.31504
5.	15249.	0.15709	0.22658
6.	18299.	0.17227	0.10829
7.	20828.	0.22243	0.

Calculate Rejection Probability for Each Stage

In this step, we calculate stage-by-stage rejection probabilities using Epstein, et al. 1963, equation 22:

$$\sum_{stage=1}^{n} (ContinuationProbability[stage - 1, trueMTBF] - ContinuationProbability[stage, trueMTBF] - AcceptanceProbability[stage, trueMTBF])$$

- **Cumulative Rejection Probabilities for Each Stage When *trueMTBF* is Symbolic**

The cumulative rejection probability for stage one when *trueMTBF* is left symbolic is:

$$\sum_{stage=1}^{1} (ContinuationProbability[stage - 1, trueMTBF] - ContinuationProbability[stage, trueMTBF] - AcceptanceProbability[stage, trueMTBF])$$

$$1 - e^{-3050/trueMTBF} - \frac{4651250\, e^{-3050/trueMTBF}}{trueMTBF^2} - \frac{3050\, e^{-3050/trueMTBF}}{trueMTBF}$$

This is an exact, but partially symbolic result. An exact result can be obtained for a specific value of *trueMTBF* such as the lower-test MTBF as follows:

```
% /. trueMTBF → 2200
```

$$1 - \frac{12961}{3872\, e^{61/44}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.163216544765922154593 4
```

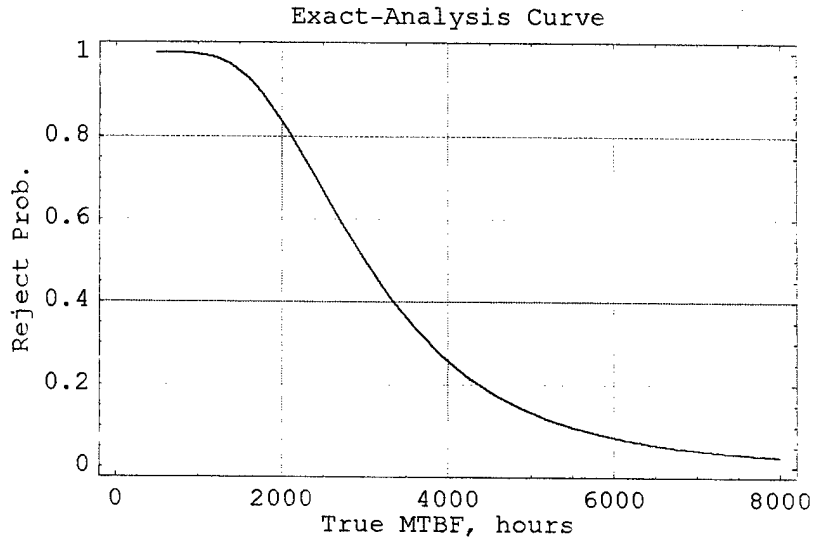
Next, the cumulative rejection probability for stage seven when *trueMTBF* is symbolic will be generated. This is one minus the operational-characteristic function.

```
rejfunction = Sum[(ContinuationProbability[stage - 1, trueMTBF] -
    ContinuationProbability[stage, trueMTBF] -
    AcceptanceProbability[stage, trueMTBF]), {stage, 1, 7}]
```

$$1 - e^{-6100/\text{trueMTBF}} - \frac{47979080106533936795875 0\, e^{-20828/\text{trueMTBF}}}{9\ \text{trueMTBF}^6} - \frac{2757330074746755125 0\, e^{-20828/\text{trueMTBF}}}{3\ \text{trueMTBF}^5} - \frac{1456352673542500\, e^{-18299/\text{trueMTBF}}}{\text{trueMTBF}^4} - \frac{695017682500\, e^{-15249/\text{trueMTBF}}}{3\ \text{trueMTBF}^3} - \frac{37210000\, e^{-12199/\text{trueMTBF}}}{\text{trueMTBF}^2} - \frac{6100\, e^{-9150/\text{trueMTBF}}}{\text{trueMTBF}}$$

rejfunction provides the exact rejection probability as a function of *trueMTBF*. This function can now be plotted:

```
Plot[rejfunction, {trueMTBF, 500, 8000}, GridLines → Automatic,
  Frame → True, FrameLabel → {"True MTBF, hours",
    "Reject Prob.", "Exact-Analysis Curve" , None}];
```



- **Cumulative Rejection Probabilities for Each Stage When *trueMTBF* Equals the Upper-Test MTBF**

It would be useful to generate a list of cumulative rejection probabilities for all seven stages when the true MTBF equals the upper-test MTBF. The parameter *trueMTBF* will be left symbolic in order to be consistent with up-front calculations. The desired list is generated but display of the output is temporarily suppressed.

```
mycumrej = Table[ Sum_{stage=1}^{stagelim} (ContinuationProbability[stage - 1, trueMTBF] -
      ContinuationProbability[stage, trueMTBF] -
      AcceptanceProbability[stage, trueMTBF]), {stagelim, 1, 7}];
```

Now a table is generated which displays our calculations. A rule is used to replace *trueMTBF* with the upper-test MTBF in the cumulative rejection probabilities stored in the list *mycumrej*. The cumulative acceptance and continuation probabilities for this case are also provided for reference.

```
utMTBFtable = NumberForm[
  TableForm[{N[timeValues, 4], N[mycumacc /. trueMTBF → 4400, 4],
    N[mycumcon /. trueMTBF → 4400, 4], N[mycumrej /. trueMTBF → 4400, 4]},
   TableDirections -> {Row, Column}, TableHeadings ->
    {{"Time", "Σ Accept Pr.", "Σ Continue Pr.", "Σ Reject Pr."},
     Automatic}, TableAlignments -> Center], {6, 5}]
```

	Time	Σ Accept Pr.	Σ Continue Pr.	Σ Reject Pr.
1.	3050.	0.	0.96668	0.03332
2.	6100.	0.24998	0.68394	0.06608
3.	9150.	0.42326	0.48583	0.09091
4.	12199.	0.5434	0.34752	0.10909
5.	15249.	0.62839	0.24936	0.12225
6.	18299.	0.6891	0.15182	0.15907
7.	20828.	0.80273	0.	0.19727

Inspection of the table above reveals that each row sums to one as it must. The table of stage-by-stage accept, continue and reject probabilities is a key test-design graphic, thus it was assigned as the value of the symbol *utMTBFtable* so it could be readily inserted in Chapter 4.

In order to calculate just the final cumulative rejection probability, we can use the *rejfunction* from the previous section and employ a rule to replace *trueMTBF* with the upper-test MTBF.

```
rejfunction /. trueMTBF → 4400
```

$$1 - \frac{61364245158684439597}{4749587251200000000\, e^{5207/1100}} - \frac{582541069417}{149923840000\, e^{18299/4400}} - \frac{278007073}{102220800\, e^{15249/4400}} - \frac{3721}{1936\, e^{1109/400}} - \frac{61}{44\, e^{183/88}} - \frac{1}{e^{61/44}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.1972729380092335932949
```

This is the producer risk.

- **Cumulative Rejection Probabilities for Each Stage When *trueMTBF* Equals the Lower-Test MTBF**

It would be useful to generate a table of rejection acceptance probabilities for all seven stages when the true MTBF equals the lower-test MTBF. The list *mycumrej*, which was generated in the previous subsection, can be used for this purpose. A rule is used to replace *trueMTBF* with the lower-test MTBF in the cumulative rejection probability expressions stored in the list *mycumrej*. The cumulative acceptance and continuation probabilities for this case are also provided for reference.

```
ltMTBFtable = NumberForm[
  TableForm[{N[timeValues, 4], N[mycumacc /. trueMTBF → 2200, 4],
    N[mycumcon /. trueMTBF → 2200, 4], N[mycumrej /. trueMTBF → 2200, 4]},
   TableDirections -> {Row, Column}, TableHeadings ->
    {{"Time", "Σ Accept Pr.", "Σ Continue Pr.", "Σ Reject Pr."},
     Automatic}, TableAlignments -> Center], {6, 5}]
```

	Time	Σ Accept Pr.	Σ Continue Pr.	Σ Reject Pr.
1.	3050.	0.	0.83678	0.16322
2.	6100.	0.06249	0.60776	0.32975
3.	9150.	0.10581	0.43781	0.45638
4.	12199.	0.13584	0.31504	0.54912
5.	15249.	0.15709	0.22658	0.61632
6.	18299.	0.17227	0.10829	0.71944
7.	20828.	0.22243	0.	0.77757

Each row sums to one as it should. The table is assigned as the value of the symbol *ltMTBFtable* so that it can be easily inserted in Chapter 4.

In order to calculate just the final cumulative rejection probability, we could use the `rejfunction` from the previous section and use a rule to replace *trueMTBF* with the lower-test MTBF.

```
rejfunction /. trueMTBF → 2200
```

$$1 - \frac{48129060799900014997}{74212300800000000\, e^{5207/550}} - \frac{582541069417}{9370240000\, e^{18299/2200}} - \frac{278007073}{12777600\, e^{15249/2200}} - \frac{3721}{484\, e^{1109/200}} - \frac{61}{22\, e^{183/44}} - \frac{1}{e^{61/22}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.7775699735173586934378
```

The consumer risk equals one minus the rejection probability when the true MTBF equals the lower-test MTBF. This is one minus the answer above:

```
1 - %
```

```
0.2224300264826413065622
```

Calculate Expected Quantity of Failures and Test Time

First, we need to define a function for the probability that the test will terminate with an accept decision at a specified number of failures [Epstein, et al. 1963, equation 33]. First the general case and then the special case:

```
AccProbabilityF[failure_Integer, trueMTBF_] :=
 Module[{stage = 1}, While[failure > a[stage], stage++]; Which[
    failure > Last[Last[accept]], 0, 0 ≤ failure ≤ Last[Last[accept]],
    AcceptanceProbability[stage, failure, trueMTBF]]] /;
  failure ≤ a[Length[timeValues]]

AccProbabilityF[failure_Integer, trueMTBF_] :=
 0 /; failure > a[Length[timeValues]]
```

Now, we will define a function for the probability that the test will terminate with a reject decision at a specified number of failures [Epstein, et al. 1963 equation 34]:

```
RejProbabilityF[failure_Integer, trueMTBF_] :=
 Module[{rejectlist}, rejectlist =
   Select[Table[{stage, r[stage]}, {stage, 1, Length[timeValues]}],
     #[[2]] == failure &] /. {st_Integer, rej_Integer} → st;
  Which[Length[rejectlist] == 0, 0, Length[rejectlist] > 0,
   Sum[(ContinuationProbability[stage - 1, trueMTBF] -
      ContinuationProbability[stage, trueMTBF] -
      AcceptanceProbability[stage, trueMTBF]),
    {stage, First[rejectlist], Last[rejectlist]}]]]
```

The probability that the test will terminate with zero failures and a reject decision is:

```
RejProbabilityF[0, trueMTBF]
```

```
0
```

This is obviously correct since the first path to rejection is if three failures occur. The probability that the test will terminate with one or two failures and a reject decision must also be zero:

```
RejProbabilityF[1, trueMTBF]
```

```
0
```

```
RejProbabilityF[2, trueMTBF]
```

```
0
```

The probability that the test will terminate with three failures and a reject decision is:

```
RejProbabilityF[3, trueMTBF]
```

$$1 - e^{-3050/\text{trueMTBF}} - \frac{4651250\, e^{-3050/\text{trueMTBF}}}{\text{trueMTBF}^2} - \frac{3050\, e^{-3050/\text{trueMTBF}}}{\text{trueMTBF}}$$

If *trueMTBF* is equal to the lower-test MTBF, the probability that the test will terminate with three failures and a reject decision is:

```
% /. trueMTBF → 2200
```

$$1 - \frac{12961}{3872\, e^{61/44}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
0.163216544765922154593 4
```

Now, we will define a function for the probability that the test will terminate with a specified number of failures [Epstein, et al. 1963 equation 35]:

```
TerminateProbability[failure_Integer, trueMTBF_] :=
 AccProbabilityF[failure, trueMTBF] + RejProbabilityF[failure, trueMTBF]
```

The probability that the test will terminate with zero failures is:

```
TerminateProbability[0, trueMTBF]
```

$$e^{-6100/trueMTBF}$$

The probability that the test will terminate with between zero and seven failures is:

```
Sum[TerminateProbability[failure, trueMTBF], {failure, 0, 7}]
```

$$1$$

This result is obviously correct since it's not possible for the test to continue beyond the seventh failure.

Next, we will define a function for the expected termination failure quantity [Epstein, et al. 1963 equation 36]:

```
ExpectedTerminationFailure[trueMTBF_] :=
 Sum[failure TerminateProbability[failure, trueMTBF],
  {failure, 0, r[Length[timeValues]]}]
```

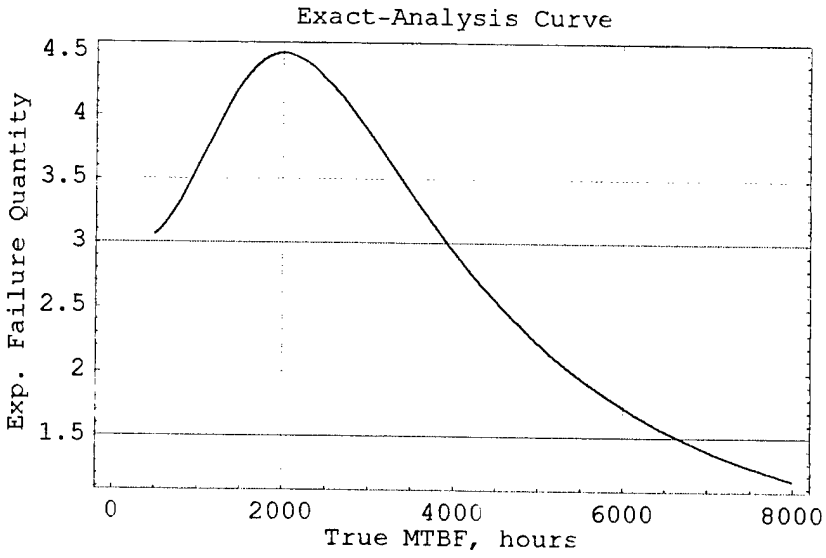
A function for the expected termination failure quantity with *trueMTBF* left symbolic is:

expectedfailurefunction = ExpectedTerminationFailure[trueMTBF]

$$
\begin{aligned}
&7\left(-\frac{4797908010653393679587 50\, e^{-20828/\text{trueMTBF}}}{9\,\text{trueMTBF}^6}-\right.\\
&\quad\frac{27573300747467551250\, e^{-20828/\text{trueMTBF}}}{3\,\text{trueMTBF}^5}+\\
&\quad\frac{4154747817367926250\, e^{-12199/\text{trueMTBF}}}{3\,\text{trueMTBF}^5}-\\
&\quad\frac{1456352673542500\, e^{-18299/\text{trueMTBF}}}{\text{trueMTBF}^4}+\frac{2249254089002500\, e^{-12199/\text{trueMTBF}}}{3\,\text{trueMTBF}^4}-\\
&\quad\left.\frac{695017682500\, e^{-15249/\text{trueMTBF}}}{3\,\text{trueMTBF}^3}+\frac{695017682500\, e^{-12199/\text{trueMTBF}}}{3\,\text{trueMTBF}^3}\right)+\\
&6\left(\frac{4797908010653393679587 50\, e^{-20828/\text{trueMTBF}}}{9\,\text{trueMTBF}^6}-\right.\\
&\quad\frac{4154747817367926250\, e^{-12199/\text{trueMTBF}}}{3\,\text{trueMTBF}^5}-\\
&\quad\frac{2249254089002500\, e^{-12199/\text{trueMTBF}}}{3\,\text{trueMTBF}^4}+\frac{216341265625000\, e^{-9150/\text{trueMTBF}}}{\text{trueMTBF}^4}-\\
&\quad\frac{695017682500\, e^{-12199/\text{trueMTBF}}}{3\,\text{trueMTBF}^3}+\frac{354657812500\, e^{-9150/\text{trueMTBF}}}{3\,\text{trueMTBF}^3}-\\
&\quad\left.\frac{37210000\, e^{-12199/\text{trueMTBF}}}{\text{trueMTBF}^2}+\frac{37210000\, e^{-9150/\text{trueMTBF}}}{\text{trueMTBF}^2}\right)+\\
&5\left(\frac{27573300747467551250\, e^{-20828/\text{trueMTBF}}}{3\,\text{trueMTBF}^5}-\frac{216341265625000\, e^{-9150/\text{trueMTBF}}}{\text{trueMTBF}^4}-\right.\\
&\quad\frac{354657812500\, e^{-9150/\text{trueMTBF}}}{3\,\text{trueMTBF}^3}+\frac{99304187500\, e^{-6100/\text{trueMTBF}}}{3\,\text{trueMTBF}^3}-\\
&\quad\frac{37210000\, e^{-9150/\text{trueMTBF}}}{\text{trueMTBF}^2}+\frac{18605000\, e^{-6100/\text{trueMTBF}}}{\text{trueMTBF}^2}-\\
&\quad\left.\frac{6100\, e^{-9150/\text{trueMTBF}}}{\text{trueMTBF}}+\frac{6100\, e^{-6100/\text{trueMTBF}}}{\text{trueMTBF}}\right)+\\
&3\left(1-e^{-3050/\text{trueMTBF}}+\frac{695017682500\, e^{-15249/\text{trueMTBF}}}{3\,\text{trueMTBF}^3}-\right.\\
&\quad\left.\frac{4651250\, e^{-3050/\text{trueMTBF}}}{\text{trueMTBF}^2}-\frac{3050\, e^{-3050/\text{trueMTBF}}}{\text{trueMTBF}}\right)+\\
&4\left(-e^{-6100/\text{trueMTBF}}+e^{-3050/\text{trueMTBF}}+\frac{1456352673542500\, e^{-18299/\text{trueMTBF}}}{\text{trueMTBF}^4}-\right.\\
&\quad\frac{99304187500\, e^{-6100/\text{trueMTBF}}}{3\,\text{trueMTBF}^3}-\frac{18605000\, e^{-6100/\text{trueMTBF}}}{\text{trueMTBF}^2}+\\
&\quad\left.\frac{4651250\, e^{-3050/\text{trueMTBF}}}{\text{trueMTBF}^2}-\frac{6100\, e^{-6100/\text{trueMTBF}}}{\text{trueMTBF}}+\frac{3050\, e^{-3050/\text{trueMTBF}}}{\text{trueMTBF}}\right)+\\
&\frac{74420000\, e^{-12199/\text{trueMTBF}}}{\text{trueMTBF}^2}+\frac{6100\, e^{-9150/\text{trueMTBF}}}{\text{trueMTBF}}
\end{aligned}
$$

Now we can plot this function:

```
expectedfailuresPlot = Plot[expectedfailurefunction,
   {trueMTBF, 500, 8000}, GridLines → Automatic,
   Frame → True, FrameLabel → {"True MTBF, hours",
     "Exp. Failure Quantity", "Exact-Analysis Curve" , None}];
```



The plot above, since it is a key test-design graphic, is assigned as the value of the symbol *expectedfailuresPlot* so that it can be easily inserted in Chapter 4.

In order to calculate the expected failure quantity for a true MTBF equal to the lower-test MTBF, we could use *expectedfailurefunction* and a rule to replace *trueMTBF* with it.

```
expectedfailurefunction /. trueMTBF → 2200
```

$$7\left(-\frac{48129060799900014997}{74212300800000000\,e^{5207/550}}-\frac{582541069417}{9370240000\,e^{18299/2200}}-\frac{278007073}{12777600\,e^{15249/2200}}+\frac{3324531305726247}{41229056000000\,e^{1109/200}}\right)+6\left(\frac{34893876441115590397}{74212300800000000\,e^{5207/550}}-\frac{3641500969726247}{41229056000000\,e^{1109/200}}+\frac{630259259}{22488576\,e^{183/44}}\right)+5\left(\frac{22058640597974041}{123687168000000\,e^{5207/550}}-\frac{692613947}{22488576\,e^{183/44}}+\frac{4970707}{511104\,e^{61/22}}\right)+3\left(1+\frac{278007073}{12777600\,e^{15249/2200}}-\frac{12961}{3872\,e^{61/44}}\right)+4\left(\frac{582541069417}{9370240000\,e^{18299/2200}}-\frac{5481811}{511104\,e^{61/22}}+\frac{12961}{3872\,e^{61/44}}\right)+\frac{3721}{242\,e^{1109/200}}+\frac{61}{22\,e^{183/44}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
4.459530562623423959919
```

In order to calculate the expected failure quantity for a true MTBF equal to the upper-test MTBF, we could use *expectedfailurefunction* and a rule to replace *trueMTBF* with it.

```
expectedfailurefunction /. trueMTBF → 4400
```

$$7\left(-\frac{61364245158684439597}{4749587251200000000\ e^{5207/1100}}-\frac{582541069417}{149923840000\ e^{18299/4400}}-\frac{278007073}{102220800\ e^{15249/4400}}+\frac{22005606513743141}{3957989376000000\ e^{1109/400}}\right)+6\left(\frac{34893876441115590397}{4749587251200000000\ e^{5207/1100}}-\frac{29612878449743141}{3957989376000000\ e^{1109/400}}+\frac{1398615991}{359817216\ e^{183/88}}\right)+5\left(\frac{22058640597974041}{3957989376000000\ e^{5207/1100}}-\frac{1897453495}{359817216\ e^{183/88}}+\frac{11186851}{4088832\ e^{61/44}}\right)+3\left(1+\frac{278007073}{102220800\ e^{15249/4400}}-\frac{29945}{15488\ e^{61/88}}\right)+4\left(\frac{582541069417}{149923840000\ e^{18299/4400}}-\frac{15275683}{4088832\ e^{61/44}}+\frac{29945}{15488\ e^{61/88}}\right)+\frac{3721}{968\ e^{1109/400}}+\frac{61}{44\ e^{183/88}}$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
2.625630136024280527324
```

Next, we will define a function for the expected test time [Epstein, et al. 1963 equation 41]:

```
ExpectedTestTime[trueMTBF_] :=
 trueMTBF ExpectedTerminationFailure[trueMTBF]
```

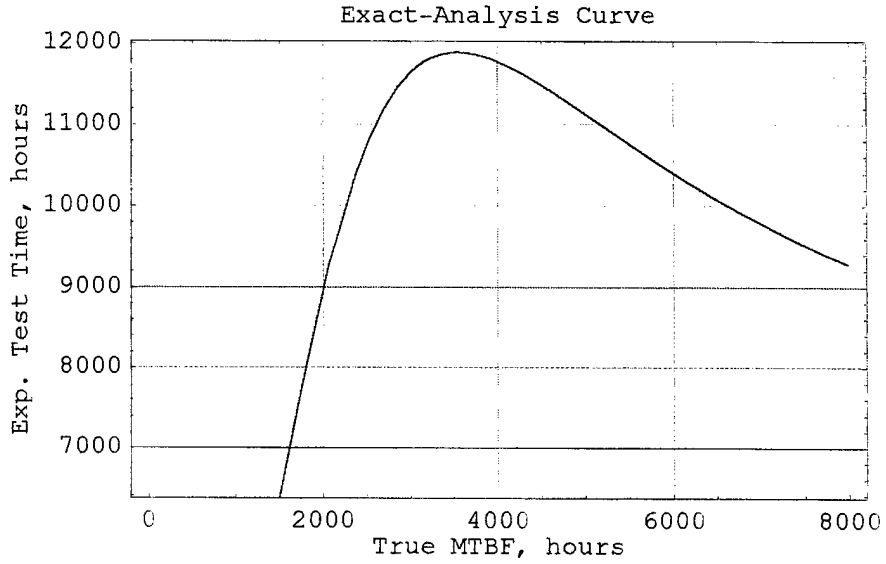
A function for the expected test time with *trueMTBF* left symbolic is:

expectedtesttimefunction = ExpectedTestTime[trueMTBF]

$$\left(7\left(-\frac{4797908010653393679587 50\, e^{-20828/trueMTBF}}{9\, trueMTBF^{6}}-\frac{27573300747467551250\, e^{-20828/trueMTBF}}{3\, trueMTBF^{5}}+\frac{4154747817367926250\, e^{-12199/trueMTBF}}{3\, trueMTBF^{5}}-\frac{1456352673542500\, e^{-18299/trueMTBF}}{trueMTBF^{4}}+\frac{2249254089002500\, e^{-12199/trueMTBF}}{3\, trueMTBF^{4}}-\frac{695017682500\, e^{-15249/trueMTBF}}{3\, trueMTBF^{3}}+\frac{695017682500\, e^{-12199/trueMTBF}}{3\, trueMTBF^{3}}\right)+6\left(\frac{4797908010653393679587 50\, e^{-20828/trueMTBF}}{9\, trueMTBF^{6}}-\frac{4154747817367926250\, e^{-12199/trueMTBF}}{3\, trueMTBF^{5}}-\frac{2249254089002500\, e^{-12199/trueMTBF}}{3\, trueMTBF^{4}}+\frac{216341265625000\, e^{-9150/trueMTBF}}{trueMTBF^{4}}-\frac{695017682500\, e^{-12199/trueMTBF}}{3\, trueMTBF^{3}}+\frac{354657812500\, e^{-9150/trueMTBF}}{3\, trueMTBF^{3}}-\frac{37210000\, e^{-12199/trueMTBF}}{trueMTBF^{2}}+\frac{37210000\, e^{-9150/trueMTBF}}{trueMTBF^{2}}\right)+5\left(\frac{27573300747467551250\, e^{-20828/trueMTBF}}{3\, trueMTBF^{5}}-\frac{216341265625000\, e^{-9150/trueMTBF}}{trueMTBF^{4}}-\frac{354657812500\, e^{-9150/trueMTBF}}{3\, trueMTBF^{3}}+\frac{99304187500\, e^{-6100/trueMTBF}}{3\, trueMTBF^{3}}-\frac{37210000\, e^{-9150/trueMTBF}}{trueMTBF^{2}}+\frac{18605000\, e^{-6100/trueMTBF}}{trueMTBF^{2}}-\frac{6100\, e^{-9150/trueMTBF}}{trueMTBF}+\frac{6100\, e^{-6100/trueMTBF}}{trueMTBF}\right)+3\left(1-e^{-3050/trueMTBF}+\frac{695017682500\, e^{-15249/trueMTBF}}{3\, trueMTBF^{3}}-\frac{4651250\, e^{-3050/trueMTBF}}{trueMTBF^{2}}-\frac{3050\, e^{-3050/trueMTBF}}{trueMTBF}\right)+4\left(-e^{-6100/trueMTBF}+e^{-3050/trueMTBF}+\frac{1456352673542500\, e^{-18299/trueMTBF}}{trueMTBF^{4}}-\frac{99304187500\, e^{-6100/trueMTBF}}{3\, trueMTBF^{3}}-\frac{18605000\, e^{-6100/trueMTBF}}{trueMTBF^{2}}+\frac{4651250\, e^{-3050/trueMTBF}}{trueMTBF^{2}}-\frac{6100\, e^{-6100/trueMTBF}}{trueMTBF}+\frac{3050\, e^{-3050/trueMTBF}}{trueMTBF}\right)+\frac{74420000\, e^{-12199/trueMTBF}}{trueMTBF^{2}}+\frac{6100\, e^{-9150/trueMTBF}}{trueMTBF}\right)trueMTBF$$

Now we can plot this function:

```
expectedtesttimePlot = Plot[expectedtesttimefunction,
   {trueMTBF, 500, 8000}, GridLines → Automatic,
   Frame → True, FrameLabel → {"True MTBF, hours",
     "Exp. Test Time, hours", "Exact-Analysis Curve" , None}];
```



The plot above, also a key test-design graphic, is assigned as the value of the symbol *expectedtesttime-Plot* so that it can be easily inserted in Chapter 4.

In order to calculate the expected test time for a true MTBF equal to the lower-test MTBF multiple, we could use *expectedtesttimefunction* and a rule to replace *trueMTBF* it.

```
expectedtesttimefunction /. trueMTBF → 2200
```

$$2200\left(7\left(-\frac{48129060799900014997}{74212300800000000\,e^{5207/550}}-\frac{582541069417}{9370240000\,e^{18299/2200}}-\frac{278007073}{12777600\,e^{15249/2200}}+\frac{3324531305726247}{41229056000000\,e^{1109/200}}\right)+6\left(\frac{34893876441115590397}{74212300800000000\,e^{5207/550}}-\frac{3641500969726247}{41229056000000\,e^{1109/200}}+\frac{630259259}{22488576\,e^{183/44}}\right)+5\left(\frac{22058640597974041}{123687168000000\,e^{5207/550}}-\frac{692613947}{22488576\,e^{183/44}}+\frac{4970707}{511104\,e^{61/22}}\right)+3\left(1+\frac{278007073}{12777600\,e^{15249/2200}}-\frac{12961}{3872\,e^{61/44}}\right)+4\left(\frac{582541069417}{9370240000\,e^{18299/2200}}-\frac{5481811}{511104\,e^{61/22}}+\frac{12961}{3872\,e^{61/44}}\right)+\frac{3721}{242\,e^{1109/200}}+\frac{61}{22\,e^{183/44}}\right)$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
9810.967237771532711822
```

In order to calculate the expected test time for a true MTBF equal to the upper-test MTBF, we could use *expectedtesttimefunction* and a rule to replace *trueMTBF* with it.

```
expectedtesttimefunction /. trueMTBF → 4400
```

$$4400\left(7\left(-\frac{61364245158684439597}{4749587251200000000\,e^{5207/1100}}-\frac{582541069417}{149923840000\,e^{18299/4400}}-\frac{278007073}{102220800\,e^{15249/4400}}+\frac{22005606513743141}{3957989376000000\,e^{1109/400}}\right)+6\left(\frac{34893876441115590397}{4749587251200000000\,e^{5207/1100}}-\frac{29612878449743141}{3957989376000000\,e^{1109/400}}+\frac{1398615991}{359817216\,e^{183/88}}\right)+5\left(\frac{22058640597974041}{3957989376000000\,e^{5207/1100}}-\frac{1897453495}{359817216\,e^{183/88}}+\frac{11186851}{4088832\,e^{61/44}}\right)+3\left(1+\frac{278007073}{102220800\,e^{15249/4400}}-\frac{29945}{15488\,e^{61/88}}\right)+4\left(\frac{582541069417}{149923840000\,e^{18299/4400}}-\frac{15275683}{4088832\,e^{61/44}}+\frac{29945}{15488\,e^{61/88}}\right)+\frac{3721}{968\,e^{1109/400}}+\frac{61}{44\,e^{183/88}}\right)$$

This is an exact result. A numerical approximation accurate to 22 decimal places is:

```
N[%, 22]
```

```
11552.77259850683432023
```

Summary

An analysis of the exact stage-by-stage acceptance, continuation and rejection probabilities (including the impact of truncation) resulting from the exponential sequential test design of Chapter 4 was performed. An exact operational-characteristic function was obtained and plotted. The consumer risk is approximately 22.2% and the producer risk is approximately 19.7%. Exact functions for expected failure quantity and test time were obtained and plotted as well.

Appendix D

Exponential Sequential Test Design Package

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix D

Exponential Sequential Test Design Package

This notebook documents the development of sequential test design functions for the Exponential distribution.

Reference

Title

Exponential Sequential Test Design Package Development

Author

John A. Sereno and Michael J. Cushing

Summary

This notebook serves two purposes. First it documents the development of the *Exponential Sequential Test Design* package. Second, it serves as the master for the *Exponential Sequential Test Design* package (i.e., it automatically generates the package).

Copyright

Notebook Version

0.8.0

Mathematica Version

3.0

History

This notebook was developed to provide functions and graphical plotting which allow for an automated capability for designing exponential sequential tests.

Keywords

reliability, exponential sequential test design, Exponential distribution, operational-characteristic curve, sequential probability ratio test, probability ratio sequential test

Source

Aroian, L.A., "Sequential analysis, direct method", *Technometrics*, vol. 10, no. 1, pp. 125-132, Feb. 1968.

Aroian, L.A., Robinson, D.E., "Direct methods for exact truncated sequential tests of the mean of a Normal distribution", *Technometrics*, vol. 11, no. 4, pp. 661-675, 1969.

Dvoretsky, A., Kiefer, J., Wolfowitz, J., "Sequential decision problems for processes with continuous time parameter: Testing hypotheses", Annals of Mathematical Statistics, vol. 24, pp. 254-264, 1953.

Epstein, B., Sobel, M., "Sequential life tests in the exponential case", *Annals of Mathematical Statistics*, vol. 26, pp. 82-93, 1955.

Epstein, B., "Truncated life tests in the exponential case", *Annals of Mathematical Statistics*, vol. 25, pp. 555-564, 1955.

Epstein, B., Patterson, A., Qualls, C., "The exact analysis of sequential life tests with particular application to AGREE plans", *Proc. Joint AIAA-SAE-ASME Aerospace Reliability and Maintainability Conference*, May 1963, pp. 284-311.

Kapur, K. & Lamberson, L. *Reliability in Engineering Design*, John Wiley & Sons, 1977.

Kececioglu, D., *Reliability & Life Testing Handbook Vol 2*, PTR Prentice Hall, 1994.

Maeder, R., *Programming in Mathematica*, 3rd ed. Addison-Wesley, 1996.

Rao, S. *Reliability-Based Design*, McGraw-Hill Inc, 1992.

US Military Handbook 781, *Handbook for Reliability Test Methods, Plans and Environments for Engineering, Development, Qualification and Production*, version A, Apr 1996.

Wald, A., "Sequential tests of statistical hypotheses", *Annals of Mathematical Statistics* , June 1945, pp. 117-186.

Wald, A., *Sequential Analysis*, John Wiley & Sons, 1947.

Warnings

Note: all cells marked as "InitializationCell" will be written to the Auto-Save package. This package can then be read in programs that use it with `Needs["Template`"]`. Cells not intended to belong to the package do not have this property.

Limitation

At present, the impact of truncation cannot be analyzed using the functions in this package. We have stand-alone examples which implement the exact-analysis method of Epstein, Patterson and Qualls [1963] in *Mathematica*. We also have stand-alone, *Mathematica*-based simulation examples.

Discussion

None.

Requirements

The package `Utilities`FilterOptions`` is used by this package.

Interface

This section declares the publicly visible functions, options, and values.

- **Set up the package context, including public imports**

```
BeginPackage["Reliability`ExponentialSequentialTestDesign`",
 "Utilities`FilterOptions`"]
```

- **Usage messages for the exported functions and the context itself**

```
ExponentialSequentialTestDesign::"usage" =
 "ExponentialSequentialTestDesign.m (version 0.8.0) is a
   package which contains a collection of functions useful for
   sequential test design based on the Exponential distribution."

ExponentialAccept::"usage" =
 "ExponentialAccept[ltmtbf,β,utmtbf,α,time] provides cumulative
   failures for acceptance as a function of cumulative test
   time given specified values for lower-test MTBF, β (
   consumer risk), upper-test MTBF and α (producer risk)."

ExponentialReject::"usage" =
 "ExponentialReject[ltmtbf,β,utmtbf,α,time,opts] provides
   cumulative failures for rejection as a function of cumulative
   test time given specified values for lower-test MTBF, β (
   consumer risk), upper-test MTBF and α (producer risk)."
```

```
ExponentialAcceptProbability::"usage" =
 "ExponentialAcceptProbability[ltmtbf,β,utmtbf,α,h,opts] calculates
   the acceptance probability given specified values for lower-
   test MTBF, β (consumer risk), upper-test MTBF, α (producer
   risk) and the exponent h. ExponentialAcceptProbability[
   ltmtbf,β,utmtbf,α,truemtbf,startpt,opts] calculates the
   acceptance probability given specified values for lower-
   test MTBF, β, upper-test MTBF, α, true MTBF and startpt (
   FindRoot starting point for the exponent h). This function
   provides approximate results in the non-truncated case."

ExponentialExactAcceptProbability::"usage" =
 "ExponentialExactAcceptProbability[ltmtbf,β,utmtbf,α,
   truemtbf,a] calculates the exact acceptance probability
   for the non-truncated case given specified values for
   lower-test MTBF, β (consumer risk), upper-test MTBF, α (
   producer risk), true MTBF and the constant a (Log[A])."

ExponentialTrueMTBF::"usage" =
 "ExponentialTrueMTBF[ltmtbf,utmtbf,h] is an implicit equation which
   relates the true MTBF to the lower-test MTBF, upper-test MTBF
   and the exponent h. It is approximate in the non-truncated case
   and is used in conjunction with ExponentialAcceptProbability."

ExponentialExpectedFailures::"usage" =
 "ExponentialExpectedFailures[ltmtbf,β,utmtbf,α,truemtbf,
   startpt,opts] calculates the expected quantity of failures
   given specified values for lower-test MTBF, β (consumer
   risk), upper-test MTBF, α (producer risk), true MTBF and
   startpt (FindRoot starting point for the exponent h). It
   provides approximate results in the non-truncated case."

ExponentialTruncationFailures::"usage" =
 "ExponentialTruncationFailures[ltmtbf,β,utmtbf,α] calculates
   the quantity of failures which would result in a truncated
   reject decision given specified values for lower-test MTBF,
   β (consumer risk), upper-test MTBF and α (producer risk)."

ExponentialTruncationTime::"usage" =
 "ExponentialTruncationTime[utmtbf,α,failures] calculates
   the quantity of time which would result in a truncated
   accept decision given specified values for upper-test
   MTBF, α (producer risk) and truncation failures."
```

```
ConstantAMethod::"usage" =
 "ConstantAMethod is an option for specifying the approximation
   method for the constant A that appears in the reject-line
   intercept. Alternatively, a numerical value can be supplied for A."

Epstein::"usage" =
 "Epstein is a choice for the option ConstantAMethod.
   ConstantAMethod -> Epstein causes the Epstein and
   Sobel approximation to be used for the constant A."

Wald::"usage" = "Wald is a choice for the option
   ConstantAMethod. ConstantAMethod -> Wald causes the Epstein
   and Sobel approximation to be used for the constant A."
```

- **Error messages for the exported objects**

See subsections of each of the exported functions for a discussion of error trapping and messages.

ExponentialAccept

```
ExponentialAccept::"comparg" = "No arguments may be complex numbers."

ExponentialAccept::"rangearg" =
 "No MTBF or risk values may be ≤ 0 and no risk values may be ≥ 1."

ExponentialAccept::"revrel" =
 "Lower-test MTBF may not be ≥ upper-test MTBF."

ExponentialAccept::"possym" =
 "Test length is expected to be either symbolic or positive."
```

ExponentialReject

```
ExponentialReject::"comparg" = "No arguments may be complex numbers."

ExponentialReject::"rangearg" =
 "No MTBF or risk values may be ≤ 0 and no risk values may be ≥ 1."

ExponentialReject::"revrel" =
 "Lower-test MTBF may not be ≥ upper-test MTBF."
```

```
ExponentialReject::"possym" =
 "Test length is expected to be either symbolic or positive."
```

ConstantAMethod

```
ConstantAMethod::"methopt" =
 "Method setting is expected to be either Wald or Epstein.
   The default setting for this option will be used instead."
```

ExponentialTrueMTBF

```
ExponentialTrueMTBF::"comparg" = "No arguments may be complex numbers."

ExponentialTrueMTBF::"rangearg" =
 "No MTBF or risk values may be ≤ 0 and no risk values may be ≥ 1."

ExponentialTrueMTBF::"revrel" =
 "Lower-test MTBF may not be ≥ upper-test MTBF."
```

ExponentialAcceptProbability

```
ExponentialAcceptProbability::"comparg" =
 "No arguments may be complex numbers."

ExponentialAcceptProbability::"rangearg" =
 "No MTBF or risk values may be ≤ 0 and no risk values may be ≥ 1."

ExponentialAcceptProbability::"revrel" =
 "Lower-test MTBF may not be ≥ upper-test MTBF."

ExponentialAcceptProbability::"frns" = "The numerical
   root-finding starting point is expected to be numeric."
```

ExponentialExactAcceptProbability

```
ExponentialExactAcceptProbability::"comparg" =
 "No arguments may be complex numbers."

ExponentialExactAcceptProbability::"rangearg" =
 "No MTBF or risk values may be ≤ 0 and no risk values may be ≥ 1."
```

```
ExponentialExactAcceptProbability::"revrel" =
 "Lower-test MTBF may not be ≥ upper-test MTBF."
```

ExponentialExpectedFailures

```
ExponentialExpectedFailures::"comparg" =
 "No arguments may be complex numbers."

ExponentialExpectedFailures::"rangearg" =
 "No MTBF or risk values may be ≤ 0 and no risk values may be ≥ 1."

ExponentialExpectedFailures::"revrel" =
 "Lower-test MTBF may not be ≥ upper-test MTBF."

ExponentialExpectedFailures::"frns" = "The numerical
   root-finding starting point is expected to be numeric."
```

ExponentialTruncationFailures

```
ExponentialTruncationFailures::"comparg" =
 "No arguments may be complex numbers."

ExponentialTruncationFailures::"rangearg" =
 "No MTBF or risk values may be ≤ 0 and no risk values may be ≥ 1."

ExponentialTruncationFailures::"revrel" =
 "Lower-test MTBF may not be ≥ upper-test MTBF."
```

ExponentialTruncationTime

```
ExponentialTruncationTime::"comparg" =
 "No arguments may be complex numbers."

ExponentialTruncationTime::"rangearg" =
 "No MTBF or risk values may be ≤ 0 and no risk values may be ≥ 1."

ExponentialTruncationTime::"possym" = "Test length is
   expected to be either symbolic or a positive integer."

ExponentialTruncationTime::"possym" =
 "Test length is expected to be either symbolic or positive."
```

Implementation

This part contains the actual definitions and any auxiliary functions that should not be visible outside.

- **Begin the private context (implementation part)**

```
Begin["`Private`"]
```

- **Unprotect any system functions for which definitions will be made**

Not applicable.

- **Definition of auxiliary functions and local (static) variables**

aK

```
aK[β_, α_] := (1 - β)/α
```

```
aK[ltmtbf_, β_, utmtbf_, α_] := ((1 - β) (utmtbf/ltmtbf + 1))/(α (2 utmtbf)/ltmtbf)
```

bK

```
bK[β_, α_] := β/(1 - α)
```

denom

```
denom[ltmtbf_, utmtbf_] := Log[utmtbf/ltmtbf]
```

hsub0

```
hsub0[ltmtbf_, β_, utmtbf_, α_] := Log[bK[β, α]]/denom[ltmtbf, utmtbf]
```

hsub1

```
hsub1[ltmtbf_, β_, utmtbf_, α_, method_] :=
 Which[method === Wald, Log[aK[β, α]],
   method === Epstein, Log[aK[ltmtbf, β, utmtbf, α]],
   NumericQ[method], Log[method]] / denom[ltmtbf, utmtbf]
```

slope

$$\text{slope}[\text{ltmtbf\_}, \text{utmtbf\_}] := \frac{\frac{1}{\text{ltmtbf}} - \frac{1}{\text{utmtbf}}}{\text{denom}[\text{ltmtbf}, \text{utmtbf}]}$$

trueMTBF

$$\text{trueMTBF}[\text{ltmtbf\_}, \text{utmtbf\_}, h\_] := \frac{\left(\frac{\text{utmtbf}}{\text{ltmtbf}}\right)^{h} - 1}{h\left(\frac{1}{\text{ltmtbf}} - \frac{1}{\text{utmtbf}}\right)}$$

acceptProb

$$\text{acceptProb}[\text{ltmtbf\_}, \beta\_, \text{utmtbf\_}, \alpha\_, h\_] := \frac{(\text{aK}[\beta, \alpha])^{h} - 1}{(\text{aK}[\beta, \alpha])^{h} - (\text{bK}[\beta, \alpha])^{h}}$$

```
acceptProb[ltmtbf_, β_, utmtbf_, α_, truemtbf_, startpt_, opts___] :=
 acceptProb[ltmtbf, β, utmtbf, α, h] /.
  FindRoot[trueMTBF[ltmtbf, utmtbf, h] == truemtbf, {h, startpt}, opts]
```

epdenom

$$\text{epdenom}[\text{ltmtbf\_}, \text{utmtbf\_}] := \frac{1}{\text{ltmtbf}} - \frac{1}{\text{utmtbf}}$$

eph0

$$\text{eph0}[\text{ltmtbf\_}, \beta\_, \text{utmtbf\_}, \alpha\_] := \frac{-\text{Log}[\text{bK}[\beta, \alpha]]}{\text{epdenom}[\text{ltmtbf}, \text{utmtbf}]}$$

eph1

$$\text{eph1}[\text{ltmtbf\_}, \beta\_, \text{utmtbf\_}, \alpha\_] := \frac{\text{Log}[\text{aK}[\beta, \alpha]]}{\text{epdenom}[\text{ltmtbf}, \text{utmtbf}]}$$

eps

$$\text{eps}[\text{ltmtbf\_}, \text{utmtbf\_}] := \frac{\text{Log}\left[\frac{\text{utmtbf}}{\text{ltmtbf}}\right]}{\text{epdenom}[\text{ltmtbf}, \text{utmtbf}]}$$

■ Definition of the exported functions

ExponentialAccept

```
ExponentialAccept[ltmtbf_, β_, utmtbf_, α_, time_] :=
 hsub0[ltmtbf, β, utmtbf, α] + time slope[ltmtbf, utmtbf] /;
  If[FreeQ[N[ltmtbf] && N[β] && N[utmtbf] && N[α] && N[time], Complex],
    True, Message[ExponentialAccept::"comparg"]; False] &&
   If[! (TrueQ[N[ltmtbf] ≥ N[utmtbf]]), True,
    Message[ExponentialAccept::"revrel"]; False] &&
   If[! (TrueQ[N[ltmtbf] ≤ 0 || N[β] ≤ 0 || N[β] ≥ 1 || N[utmtbf] ≤ 0 || N[α] ≤ 0 ||
        N[α] ≥ 1]), True, Message[ExponentialAccept::"rangearg"];
    False] && If[Positive[time] || ! (FreeQ[time, _Symbol]),
    True, Message[ExponentialAccept::"possym"]; False]
```

ExponentialReject

```
Options[ExponentialReject] = {ConstantAMethod → Epstein}

ExponentialReject[ltmtbf_, β_, utmtbf_, α_, time_, opts___] :=
 Module[{kAMethod}, kAMethod =
    ConstantAMethod /. Flatten[{opts}] /. Options[ExponentialReject];
   If[kAMethod =!= Wald && kAMethod =!= Epstein && ! NumericQ[kAMethod],
    Message[ConstantAMethod::methopt];
    kAMethod = ConstantAMethod /. Options[ExponentialReject]];
   hsub1[ltmtbf, β, utmtbf, α, kAMethod] + time slope[ltmtbf, utmtbf]] /;
  If[FreeQ[N[ltmtbf] && N[β] && N[utmtbf] && N[α] && N[time], Complex],
    True, Message[ExponentialReject::"comparg"]; False] &&
   If[! (TrueQ[N[ltmtbf] ≥ N[utmtbf]]), True,
    Message[ExponentialReject::"revrel"]; False] &&
   If[! (TrueQ[N[ltmtbf] ≤ 0 || N[β] ≤ 0 || N[β] ≥ 1 || N[utmtbf] ≤ 0 || N[α] ≤ 0 ||
        N[α] ≥ 1]), True, Message[ExponentialReject::"rangearg"];
    False] && If[Positive[time] || ! (FreeQ[time, _Symbol]),
    True, Message[ExponentialReject::"possym"]; False]
```

ExponentialTrueMTBF

```
ExponentialTrueMTBF[ltmtbf_, utmtbf_, h_] :=
 trueMTBF[ltmtbf, utmtbf, h] /;
  If[FreeQ[N[ltmtbf] && N[utmtbf] && N[h], Complex], True,
    Message[ExponentialTrueMTBF::"comparg"]; False] &&
   If[! (TrueQ[N[ltmtbf] ≥ N[utmtbf]]), True,
    Message[ExponentialTrueMTBF::"revrel"]; False] &&
   If[! (TrueQ[N[ltmtbf] ≤ 0 || N[utmtbf] ≤ 0]), True,
    Message[ExponentialTrueMTBF::"rangearg"]; False]
```

ExponentialAcceptProbability

```
Options[ExponentialAcceptProbability] = Options[FindRoot]


ExponentialAcceptProbability[ltmtbf_, β_, utmtbf_, α_, h_] :=
 acceptProb[ltmtbf, β, utmtbf, α, h] /;
  If[FreeQ[N[ltmtbf] && N[β] && N[utmtbf] && N[α] && N[h], Complex], True,
    Message[ExponentialAcceptProbability::"comparg"]; False] &&
   If[! (TrueQ[N[ltmtbf] ≥ N[utmtbf]]), True,
    Message[ExponentialAcceptProbability::"revrel"]; False] &&
   If[! (TrueQ[N[ltmtbf] ≤ 0 || N[β] ≤ 0 || N[β] ≥ 1 ||
        N[utmtbf] ≤ 0 || N[α] ≤ 0 || N[α] ≥ 1]), True,
    Message[ExponentialAcceptProbability::"rangearg"]; False]


ExponentialAcceptProbability[ltmtbf_,
  β_, utmtbf_, α_, truemtbf_, startpt_, opts___] :=
 acceptProb[ltmtbf, β, utmtbf, α, truemtbf, startpt, opts] /;
  If[FreeQ[N[ltmtbf] && N[β] && N[utmtbf] &&
      N[α] && N[truemtbf] && N[startpt], Complex], True,
    Message[ExponentialAcceptProbability::"comparg"]; False] &&
   If[! (TrueQ[N[ltmtbf] ≥ N[utmtbf]]), True,
    Message[ExponentialAcceptProbability::"revrel"]; False] &&
   If[! (TrueQ[N[ltmtbf] ≤ 0 || N[β] ≤ 0 || N[β] ≥ 1 || N[utmtbf] ≤ 0 ||
        N[α] ≤ 0 || N[α] ≥ 1 || N[truemtbf] ≤ 0]), True,
    Message[ExponentialAcceptProbability::"rangearg"]; False] &&
   If[NumericQ[startpt], True,
   Message[ExponentialAcceptProbability::frns]; False]
```

ExponentialExactAcceptProbability

```
ExponentialExactAcceptProbability[ltmtbf_, β_,
```

$$utmtbf\_, \alpha\_, truemtbf\_, a\_] := \left(\left(\frac{\beta}{1-\alpha} \right)^{\frac{1}{truemtbf\left(\frac{1}{ltmtbf}-\frac{1}{utmtbf}\right)}} \right.$$

$$\left. \sum_{i=0}^{Ceiling\left[\frac{a}{Log\left[\frac{utmtbf}{ltmtbf}\right]}-1\right]} \frac{(-1)^i \left(\frac{\left(\frac{a}{Log\left[\frac{utmtbf}{ltmtbf}\right]}-i\right) Exp\left[-\frac{1}{truemtbf\left(\frac{1}{ltmtbf}-\frac{1}{utmtbf}\right)}\right]}{\frac{truemtbf\left(\frac{1}{ltmtbf}-\frac{1}{utmtbf}\right)}{Log\left[\frac{utmtbf}{ltmtbf}\right]}} \right)^i}{i!} \right) /$$

$$\left(\sum_{i=0}^{Ceiling\left[\frac{a}{Log\left[\frac{utmtbf}{ltmtbf}\right]}-\frac{Log\left[\frac{\beta}{1-\alpha}\right]}{Log\left[\frac{utmtbf}{ltmtbf}\right]}-1\right]} \frac{1}{i!} \left((-1)^i \left(\left(\left(\frac{a}{Log\left[\frac{utmtbf}{ltmtbf}\right]} - \frac{Log\left[\frac{\beta}{1-\alpha}\right]}{Log\left[\frac{utmtbf}{ltmtbf}\right]} - i \right) Exp\left[\right. \right. \right. \right.$$

$$\left. \left. \left. \left. - \frac{1}{\frac{truemtbf\left(\frac{1}{ltmtbf}-\frac{1}{utmtbf}\right)}{Log\left[\frac{utmtbf}{ltmtbf}\right]}} \right] \right) / \frac{truemtbf\left(\frac{1}{ltmtbf}-\frac{1}{utmtbf}\right)}{Log\left[\frac{utmtbf}{ltmtbf}\right]} \right) \text{^} i \right) \right) /;$$

```
If[FreeQ[N[ltmtbf] && N[β] && N[utmtbf] && N[α] && N[truemtbf] && N[a],
   Complex], True,
  Message[ExponentialExactAcceptProbability::"comparg"]; False] &&
 If[! (TrueQ[N[ltmtbf] ≥ N[utmtbf]]), True,
  Message[ExponentialExactAcceptProbability::"revrel"]; False] &&
 If[! (TrueQ[N[ltmtbf] ≤ 0 || N[β] ≤ 0 || N[β] ≥ 1 || N[utmtbf] ≤ 0 ||
      N[α] ≤ 0 || N[α] ≥ 1 || N[truemtbf] ≤ 0]), True,
  Message[ExponentialExactAcceptProbability::"rangearg"]; False]
```

ExponentialExpectedFailures

```
Options[ExponentialExpectedFailures] = Options[FindRoot]
```

```
ExponentialExpectedFailures[ltmtbf_,
  β_, utmtbf_, α_, truemtbf_, startpt_, opts___] :=
 Module[{ap}, (eph1[ltmtbf, β, utmtbf, α] - ap (eph0[ltmtbf, β, utmtbf, α] +
          eph1[ltmtbf, β, utmtbf, α])) / (eps[ltmtbf, utmtbf] - truemtbf) /.
    ap → acceptProb[ltmtbf, β, utmtbf, α, truemtbf, startpt, opts]] /;
  If[FreeQ[N[ltmtbf] && N[β] && N[utmtbf] && N[α] &&
      N[truemtbf] && N[startpt], Complex], True,
    Message[ExponentialExpectedFailures::"comparg"]; False] &&
   If[! (TrueQ[N[ltmtbf] ≥ N[utmtbf]]), True,
    Message[ExponentialExpectedFailures::"revrel"]; False] &&
   If[! (TrueQ[N[ltmtbf] ≤ 0 || N[β] ≤ 0 || N[β] ≥ 1 || N[utmtbf] ≤ 0 ||
       N[α] ≤ 0 || N[α] ≥ 1 || N[truemtbf] ≤ 0]), True,
    Message[ExponentialExpectedFailures::"rangearg"]; False] &&
   If[NumericQ[startpt], True,
   Message[ExponentialExpectedFailures::frns]; False] &&
    (eps[ltmtbf, utmtbf] ≠ truemtbf || Head[truemtbf] == Symbol)

ExponentialExpectedFailures[ltmtbf_,
  β_, utmtbf_, α_, truemtbf_, startpt_, opts___] :=
 eph0[ltmtbf, β, utmtbf, α] eph1[ltmtbf, β, utmtbf, α]
 ------------------------------------------------------ /;
                 eps[ltmtbf, utmtbf]^2
  If[FreeQ[N[ltmtbf] && N[β] && N[utmtbf] &&
      N[α] && N[truemtbf] && N[startpt], Complex], True,
    Message[ExponentialExpectedFailures::"comparg"]; False] &&
   If[! (TrueQ[N[ltmtbf] ≥ N[utmtbf]]), True,
    Message[ExponentialExpectedFailures::"revrel"]; False] &&
   If[! (TrueQ[N[ltmtbf] ≤ 0 || N[β] ≤ 0 || N[β] ≥ 1 ||
       N[utmtbf] ≤ 0 || N[α] ≤ 0 || N[α] ≥ 1 || N[truemtbf] ≤ 0]),
    True, Message[ExponentialExpectedFailures::"rangearg"];
    False] && If[NumericQ[startpt], True,
    Message[ExponentialExpectedFailures::"frns"]; False] &&
   eps[ltmtbf, utmtbf] == truemtbf
```

ExponentialTruncationFailures

```
ExponentialTruncationFailures[ltmtbf_, β_, utmtbf_, α_] :=
 Module[{df}, df = 1; While[InverseGammaRegularized[df/2, 0, α] /
     InverseGammaRegularized[df/2, 0, 1 - β] <
    ltmtbf/utmtbf, ++df]; Return[Ceiling[df/2]]] /;
  If[FreeQ[N[ltmtbf] && N[β] && N[utmtbf] && N[α], Complex], True,
    Message[ExponentialTruncationFailures::"comparg"]; False] &&
   If[! (TrueQ[N[ltmtbf] ≥ N[utmtbf]]), True,
    Message[ExponentialTruncationFailures::"revrel"]; False] &&
   If[! (TrueQ[N[ltmtbf] ≤ 0 || N[β] ≤ 0 || N[β] ≥ 1 ||
        N[utmtbf] ≤ 0 || N[α] ≤ 0 || N[α] ≥ 1]), True,
    Message[ExponentialTruncationFailures::"rangearg"]; False]
```

ExponentialTruncationTime

```
ExponentialTruncationTime[utmtbf_, α_, failures_] :=
 utmtbf InverseGammaRegularized[failures, 0, α] /;
  If[FreeQ[N[utmtbf] && N[α] && N[failures], Complex], True,
    Message[ExponentialTruncationTime::"comparg"]; False] &&
   If[! (TrueQ[N[utmtbf] ≤ 0 || N[α] ≤ 0 || N[α] ≥ 1]), True,
    Message[ExponentialTruncationTime::"rangearg"]; False] &&
   If[(Positive[failures] && IntegerQ[failures]) ||
     ! (FreeQ[failures, _Symbol]), True,
    Message[ExponentialTruncationTime::"possym"]; False]
```

- **Definitions for system functions**

Not applicable.

- **Restore protection of system symbols**

Not applicable.

- **End the private context**

```
End[]
```

Epilog

This section protects exported symbols and ends the package.

- **Protect exported symbol**

```
Protect[ExponentialAccept, ExponentialReject, ConstantAMethod, Epstein,
 Wald, ExponentialAcceptProbability, ExponentialExactAcceptProbability,
 ExponentialTrueMTBF, ExponentialExpectedFailures,
 ExponentialTruncationFailures, ExponentialTruncationTime]
```

- **End the package context**

```
EndPackage[]
```

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix E

Simulation Supplement to Chapter 5

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix E

Simulation Supplement to Chapter 5

Introduction

This appendix contains the bulk of the simulation results for the hypergeometric sequential test plan designed in chapter 5 and was executed in conjunction with it. Chapter 5 contains the simulation assuming the defect quantity equals 40. The simulations assuming the number of defective procedures in the population of 310 was 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34 and 37 are in this appendix. This appendix must be read in conjunction with Chapter 5 in order to be properly understood.

Simulation of Decision Rules

- **Assume Defect Quantity Equals 37**

If 37 of the 310 procedures are defective, then the percentage of defectives is

```
37
--- // N
310
```

```
0.119355
```

and the percentage of non-defectives is

```
1 - %
```

```
0.880645
```

The initial quantity of defective procedures is assigned as the value of the symbol *initdef*:

```
initdef = 37;
```

The pseudorandom number generating function is:

```
fun[{stage_Integer, cumdef_Integer}] :=
 {stage + 1, cumdef + Random[HypergeometricDistribution[
     sample[[stage+1]], initdef - cumdef, proc[[stage+1]]]]}
```

The function that will test the simulation against the acceptance rules is:

```
testfun[{stage_Integer, cumdef_Integer}] := accept[stage] < cumdef
```

A simulation of 25,000 hypergeometric tests is generated as follows:

```
simlist =
  Table[Length[NestWhileList[fun, {0, 0}, testfun, 1, Length[sample]]] - 1,
   {simqty}];
```

The symbol *simlist* contains a list of the stages that the simulations ended at. The quantity of terminations at each stage are assigned as the value of *simlist37*:

```
simlist37 =
  Table[Length[Select[simlist, #1 == i &]], {i, 1, Length[sample]}];
```

The simulation results are appended to the simulation file so that they can be retrieved for subsequent analysis if need be.

```
Save[simfile, simlist37]
```

These simulation results may be easily retrieved thus:

```
<< "hypersimfile3";
```

Stage-by-stage termination probabilities are:

```
TableForm[Transpose[{Range[1, 15], N[simlist37/simqty]}],
 TableHeadings → {None, {"Stage", "Termination Probability"}},
 TableAlignments → Center]
```

Stage	Termination Probability
1	0.07104
2	0.0132
3	0.00372
4	0.00096
5	0.0002
6	0.00004
7	0.
8	0.00004
9	0.
10	0.
11	0.
12	0.
13	0.
14	0.
15	0.9108

The terminations at stages 1 - 14 are acceptances. The terminations at the last stage are simulations that went the distance. Cumulative termination probabilities are:

```
TableForm[Transpose[{Range[0, 15], N[FoldList[Plus, 0, simlist37/simqty]]}],
 TableHeadings →
  {None, {"Stage", "Cumulative Termination Probability"}},
 TableAlignments → Center]
```

Stage	Cumulative Termination Probability
0	0.
1	0.07104
2	0.08424
3	0.08796
4	0.08892
5	0.08912
6	0.08916
7	0.08916
8	0.0892
9	0.0892
10	0.0892
11	0.0892
12	0.0892
13	0.0892
14	0.0892
15	1.

The stage 1 acceptance probability is:

```
PDF[HypergeometricDistribution[20, 37, 310], 0] // N
```

```
0.0721509
```

The stage 1 acceptance probability obtained by simulation is consistent with the numerical result above. The final cumulative acceptance probability is the stage 14 termination probability, i.e., approximately 9%.

- **Assume Defect Quantity Equals 34**

If 34 of the 310 procedures are defective, then the percentage of defectives is

$$\frac{34}{310} \text{ // N}$$

```
0.109677
```

and the percentage of non-defectives is

```
1 - %
```

```
0.890323
```

The initial quantity of defective procedures is assigned as the value of the symbol *initdef*:

```
initdef = 34;
```

The pseudorandom number generating function is:

```
fun[{stage_Integer, cumdef_Integer}] :=
 {stage + 1, cumdef + Random[HypergeometricDistribution[
     sample[[stage+1]], initdef - cumdef, proc[[stage+1]]]]}
```

The function that will test the simulation against the acceptance rules is:

```
testfun[{stage_Integer, cumdef_Integer}] := accept[stage] < cumdef
```

A simulation of 25,000 hypergeometric tests is generated as follows:

```
simlist = Table[
   Length[NestWhileList[fun, {0, 0}, testfun, 1, Length[sample]]] - 1,
   {simqty}];
```

The symbol *simlist* contains a list of the stages that the simulations ended at. The quantity of terminations at each stage are assigned as the value of *simlist34*:

```
simlist34 =
  Table[Length[Select[simlist, #1 == i &]], {i, 1, Length[sample]}];
```

The simulation results are appended to the simulation file so that they can be retrieved for subsequent analysis if need be.

```
Save[simfile, simlist34]
```

These simulation results may be easily retrieved thus:

```
<< "hypersimfile3";
```

Stage-by-stage termination probabilities are:

```
TableForm[Transpose[{Range[1, 15], N[simlist34/simqty]}],
 TableHeadings → {None, {"Stage", "Termination Probability"}},
 TableAlignments → Center]
```

Stage	Termination Probability
1	0.09204
2	0.01896
3	0.00552
4	0.00244
5	0.00068
6	0.00036
7	0.00008
8	0.
9	0.
10	0.
11	0.
12	0.
13	0.
14	0.
15	0.87992

The terminations at stages 1 - 14 are acceptances. The terminations at the last stage are simulations that went the distance. Cumulative termination probabilities are:

```
TableForm[Transpose[{Range[0, 15], N[FoldList[Plus, 0, simlist34/simqty]]}],
 TableHeadings →
  {None, {"Stage", "Cumulative Termination Probability"}},
 TableAlignments → Center]
```

Stage	Cumulative Termination Probability
0	0.
1	0.09204
2	0.111
3	0.11652
4	0.11896
5	0.11964
6	0.12
7	0.12008
8	0.12008
9	0.12008
10	0.12008
11	0.12008
12	0.12008
13	0.12008
14	0.12008
15	1.

The stage 1 acceptance probability is:

```
PDF[HypergeometricDistribution[20, 34, 310], 0] // N
```

```
0.0904941
```

The stage 1 acceptance probability obtained by simulation is consistent with the numerical result above. The final cumulative acceptance probability is the stage 14 termination probability, i.e., approximately 12%.

- **Assume Defect Quantity Equals 31**

If 31 of the 310 procedures are defective, then the percentage of defectives is

```
31/310 // N
```

```
0.1
```

and the percentage of non-defectives is

```
1 - %
```

```
0.9
```

The initial quantity of defective procedures is assigned as the value of the symbol *initdef*:

```
initdef = 31;
```

The pseudorandom number generating function is:

```
fun[{stage_Integer, cumdef_Integer}] :=
 {stage + 1, cumdef + Random[HypergeometricDistribution[
     sample[[stage+1]], initdef - cumdef, proc[[stage+1]]]]}
```

The function that will test the simulation against the acceptance rules is:

```
testfun[{stage_Integer, cumdef_Integer}] := accept[stage] < cumdef
```

A simulation of 25,000 hypergeometric tests is generated as follows:

```
simlist = Table[
   Length[NestWhileList[fun, {0, 0}, testfun, 1, Length[sample]]] - 1,
   {simqty}];
```

The symbol *simlist* contains a list of the stages that the simulations ended at. The quantity of terminations at each stage are assigned as the value of *simlist31*:

```
simlist31 =
  Table[Length[Select[simlist, #1 == i &]], {i, 1, Length[sample]}];
```

The simulation results are appended to the simulation file so that they can be retrieved for subsequent analysis if need be.

```
Save[simfile, simlist31]
```

These simulation results may be easily retrieved thus:

```
<< "hypersimfile3";
```

Stage-by-stage termination probabilities are:

```
TableForm[Transpose[{Range[1, 15], N[simlist31/simqty]}],
 TableHeadings → {None, {"Stage", "Termination Probability"}},
 TableAlignments → Center]
```

Stage	Termination Probability
1	0.115
2	0.0312
3	0.0098
4	0.00408
5	0.00144
6	0.00036
7	0.0002
8	0.00004
9	0.
10	0.
11	0.
12	0.
13	0.
14	0.
15	0.83788

The terminations at stages 1 - 14 are acceptances. The terminations at the last stage are simulations that went the distance. Cumulative termination probabilities are:

```
TableForm[Transpose[{Range[0, 15], N[FoldList[Plus, 0, simlist31/simqty]]}],
 TableHeadings →
  {None, {"Stage", "Cumulative Termination Probability"}},
 TableAlignments → Center]
```

Stage	Cumulative Termination Probability
0	0.
1	0.115
2	0.1462
3	0.156
4	0.16008
5	0.16152
6	0.16188
7	0.16208
8	0.16212
9	0.16212
10	0.16212
11	0.16212
12	0.16212
13	0.16212
14	0.16212
15	1.

The stage 1 acceptance probability is:

```
PDF[HypergeometricDistribution[20, 31, 310], 0] // N
```

```
0.113213
```

The stage 1 acceptance probability obtained by simulation is consistent with the numerical result above. The final cumulative acceptance probability is the stage 14 termination probability, i.e., approximately 16%.

- **Assume Defect Quantity Equals 28**

If 28 of the 310 procedures are defective, then the percentage of defectives is

```
28/310 // N
```

```
0.0903226
```

and the percentage of non-defectives is

```
1 - %

0.909677
```

The initial quantity of defective procedures is assigned as the value of the symbol *initdef*:

```
initdef = 28;
```

The pseudorandom number generating function is:

```
fun[{stage_Integer, cumdef_Integer}] :=
 {stage + 1, cumdef + Random[HypergeometricDistribution[
     sample[[stage+1]], initdef - cumdef, proc[[stage+1]]]]}
```

The function that will test the simulation against the acceptance rules is:

```
testfun[{stage_Integer, cumdef_Integer}] := accept[stage] < cumdef
```

A simulation of 25,000 hypergeometric tests is generated as follows:

```
simlist = Table[
   Length[NestWhileList[fun, {0, 0}, testfun, 1, Length[sample]]] - 1,
   {simqty}];
```

The symbol *simlist* contains a list of the stages that the simulations ended at. The quantity of terminations at each stage are assigned as the value of *simlist28*:

```
simlist28 =
  Table[Length[Select[simlist, #1 = i &]], {i, 1, Length[sample]}];
```

The simulation results are appended to the simulation file so that they can be retrieved for subsequent analysis if need be.

```
Save[simfile, simlist28]
```

These simulation results may be easily retrieved thus:

```
<< "hypersimfile3";
```

Stage-by-stage termination probabilities are:

```
TableForm[Transpose[{Range[1, 15], N[simlist28/simqty]}],
 TableHeadings → {None, {"Stage", "Termination Probability"}},
 TableAlignments → Center]
```

Stage	Termination Probability
1	0.14216
2	0.03908
3	0.01416
4	0.0066
5	0.00324
6	0.00112
7	0.0006
8	0.00044
9	0.
10	0.
11	0.
12	0.
13	0.
14	0.
15	0.7926

The terminations at stages 1 - 14 are acceptances. The terminations at the last stage are simulations that went the distance. Cumulative termination probabilities are:

```
TableForm[Transpose[{Range[0, 15], N[FoldList[Plus, 0, simlist28/simqty]]}],
 TableHeadings →
  {None, {"Stage", "Cumulative Termination Probability"}},
 TableAlignments → Center]
```

Stage	Cumulative Termination Probability
0	0.
1	0.14216
2	0.18124
3	0.1954
4	0.202
5	0.20524
6	0.20636
7	0.20696
8	0.2074
9	0.2074
10	0.2074
11	0.2074
12	0.2074
13	0.2074
14	0.2074
15	1.

The stage 1 acceptance probability is:

```
PDF[HypergeometricDistribution[20, 28, 310], 0] // N
```

```
0.141285
```

The stage 1 acceptance probability obtained by simulation is consistent with the numerical result above. The final cumulative acceptance probability is the stage 14 termination probability, i.e., approximately 21%.

- **Assume Defect Quantity Equals 25**

If 25 of the 310 procedures are defective, then the percentage of defectives is

```
25/310 // N
```

```
0.0806452
```

and the percentage of non-defectives is

```
1 - %
```

```
0.919355
```

The initial quantity of defective procedures is assigned as the value of the symbol *initdef*:

```
initdef = 25;
```

The pseudorandom number generating function is:

```
fun[{stage_Integer, cumdef_Integer}] :=
 {stage + 1, cumdef + Random[HypergeometricDistribution[
     sample[[stage+1]], initdef - cumdef, proc[[stage+1]]]]}
```

The function that will test the simulation against the acceptance rules is:

```
testfun[{stage_Integer, cumdef_Integer}] := accept[stage] < cumdef
```

A simulation of 25,000 hypergeometric tests is generated as follows:

```
simlist = Table[
   Length[NestWhileList[fun, {0, 0}, testfun, 1, Length[sample]]] - 1,
   {simqty}];
```

The symbol *simlist* contains a list of the stages that the simulations ended at. The quantity of terminations at each stage are assigned as the value of *simlist25*:

```
simlist25 =
  Table[Length[Select[simlist, #1 == i &]], {i, 1, Length[sample]}];
```

The simulation results are appended to the simulation file so that they can be retrieved for subsequent analysis if need be.

```
Save[simfile, simlist25]
```

These simulation results may be easily retrieved thus:

```
<< "hypersimfile3";
```

Stage-by-stage termination probabilities are:

$$\texttt{TableForm}\left[\texttt{Transpose}\left[\left\{\texttt{Range}[1, 15], \texttt{N}\left[\frac{\texttt{simlist25}}{\texttt{simqty}}\right]\right\}\right], \texttt{TableHeadings} \to \{\texttt{None}, \{\texttt{"Stage"}, \texttt{"Termination Probability"}\}\}, \texttt{TableAlignments} \to \texttt{Center}\right]$$

Stage	Termination Probability
1	0.17484
2	0.05476
3	0.0246
4	0.01188
5	0.00764
6	0.005
7	0.00212
8	0.00084
9	0.0006
10	0.00012
11	0.00004
12	0.
13	0.
14	0.
15	0.71756

The terminations at stages 1 - 14 are acceptances. The terminations at the last stage are simulations that went the distance. Cumulative termination probabilities are:

```
TableForm[Transpose[{Range[0, 15], N[FoldList[Plus, 0, simlist25/simqty]]}],
 TableHeadings →
  {None, {"Stage", "Cumulative Termination Probability"}},
 TableAlignments → Center]
```

Stage	Cumulative Termination Probability
0	0.
1	0.17484
2	0.2296
3	0.2542
4	0.26608
5	0.27372
6	0.27872
7	0.28084
8	0.28168
9	0.28228
10	0.2824
11	0.28244
12	0.28244
13	0.28244
14	0.28244
15	1.

The stage 1 acceptance probability is:

```
PDF[HypergeometricDistribution[20, 25, 310], 0] // N
```

```
0.175889
```

The stage 1 acceptance probability obtained by simulation is consistent with the numerical result above. The final cumulative acceptance probability is the stage 14 termination probability, i.e., approximately 28%.

- **Assume Defect Quantity Equals 22**

If 22 of the 310 procedures are defective, then the percentage of defectives is

```
22/310 // N
```

```
0.0709677
```

and the percentage of non-defectives is

```
1 - %
```

```
0.929032
```

The initial quantity of defective procedures is assigned as the value of the symbol *initdef*:

```
initdef = 22;
```

The pseudorandom number generating function is:

```
fun[{stage_Integer, cumdef_Integer}] :=
 {stage + 1, cumdef + Random[HypergeometricDistribution[
     sample[[stage+1]], initdef - cumdef, proc[[stage+1]]]]}
```

The function that will test the simulation against the acceptance rules is:

```
testfun[{stage_Integer, cumdef_Integer}] := accept[stage] < cumdef
```

A simulation of 25,000 hypergeometric tests is generated as follows:

```
simlist = Table[
   Length[NestWhileList[fun, {0, 0}, testfun, 1, Length[sample]]] - 1,
   {simqty}];
```

The symbol *simlist* contains a list of the stages that the simulations ended at. The quantity of terminations at each stage are assigned as the value of *simlist22*:

```
simlist22 =
  Table[Length[Select[simlist, #1 == i &]], {i, 1, Length[sample]}];
```

The simulation results are appended to the simulation file so that they can be retrieved for subsequent analysis if need be.

```
Save[simfile, simlist22]
```

These simulation results may be easily retrieved thus:

```
<< "hypersimfile3";
```

Stage-by-stage termination probabilities are:

```
TableForm[Transpose[{Range[1, 15], N[simlist22/simqty]}],
 TableHeadings → {None, {"Stage", "Termination Probability"}},
 TableAlignments → Center]
```

Stage	Termination Probability
1	0.22056
2	0.0766
3	0.03748
4	0.02264
5	0.01412
6	0.00892
7	0.00552
8	0.00364
9	0.00232
10	0.00084
11	0.00032
12	0.00012
13	0.00008
14	0.
15	0.60684

The terminations at stages 1 - 14 are acceptances. The terminations at the last stage are simulations that went the distance. Cumulative termination probabilities are:

```
TableForm[Transpose[{Range[0, 15], N[FoldList[Plus, 0, simlist22/simqty]]}],
 TableHeadings →
  {None, {"Stage", "Cumulative Termination Probability"}},
 TableAlignments → Center]
```

Stage	Cumulative Termination Probability
0	0.
1	0.22056
2	0.29716
3	0.33464
4	0.35728
5	0.3714
6	0.38032
7	0.38584
8	0.38948
9	0.3918
10	0.39264
11	0.39296
12	0.39308
13	0.39316
14	0.39316
15	1.

The stage 1 acceptance probability is:

```
PDF[HypergeometricDistribution[20, 22, 310], 0] // N
```

```
0.21845
```

The stage 1 acceptance probability obtained by simulation is consistent with the numerical result above. The final cumulative acceptance probability is the stage 14 termination probability, i.e., approximately 39%.

- **Assume Defect Quantity Equals 19**

If 19 of the 310 procedures are defective, then the percentage of defectives is

```
19/310 // N
```

```
0.0612903
```

and the percentage of non-defectives is

```
1 - %

0.93871
```

The initial quantity of defective procedures is assigned as the value of the symbol *initdef*:

```
initdef = 19;
```

The pseudorandom number generating function is:

```
fun[{stage_Integer, cumdef_Integer}] :=
 {stage + 1, cumdef + Random[HypergeometricDistribution[
     sample[[stage+1]], initdef - cumdef, proc[[stage+1]]]]}
```

The function that will test the simulation against the acceptance rules is:

```
testfun[{stage_Integer, cumdef_Integer}] := accept[stage] < cumdef
```

A simulation of 25,000 hypergeometric tests is generated as follows:

```
simlist = Table[
   Length[NestWhileList[fun, {0, 0}, testfun, 1, Length[sample]]] - 1,
   {simqty}];
```

The symbol *simlist* contains a list of the stages that the simulations ended at. The quantity of terminations at each stage are assigned as the value of *simlist19*:

```
simlist19 =
  Table[Length[Select[simlist, #1 == i &]], {i, 1, Length[sample]}];
```

The simulation results are appended to the simulation file so that they can be retrieved for subsequent analysis if need be.

```
Save[simfile, simlist19]
```

These simulation results may be easily retrieved thus:

```
<< "hypersimfile3";
```

Stage-by-stage termination probabilities are:

```
TableForm[Transpose[{Range[1, 15], N[simlist19/simqty]}],
 TableHeadings → {None, {"Stage", "Termination Probability"}},
 TableAlignments → Center]
```

Stage	Termination Probability
1	0.26868
2	0.103
3	0.05496
4	0.035
5	0.02404
6	0.0178
7	0.01364
8	0.00968
9	0.00712
10	0.00552
11	0.00336
12	0.00204
13	0.00116
14	0.00032
15	0.45368

The terminations at stages 1 - 14 are acceptances. The terminations at the last stage are simulations that went the distance. Cumulative termination probabilities are:

```
TableForm[Transpose[{Range[0, 15], N[FoldList[Plus, 0, simlist19/simqty]]}],
 TableHeadings →
  {None, {"Stage", "Cumulative Termination Probability"}},
 TableAlignments → Center]
```

Stage	Cumulative Termination Probability
0	0.
1	0.26868
2	0.37168
3	0.42664
4	0.46164
5	0.48568
6	0.50348
7	0.51712
8	0.5268
9	0.53392
10	0.53944
11	0.5428
12	0.54484
13	0.546
14	0.54632
15	1.

The stage 1 acceptance probability is:

```
PDF[HypergeometricDistribution[20, 19, 310], 0] // N
```

```
0.270679
```

The stage 1 acceptance probability obtained by simulation is consistent with the numerical result above. The final cumulative acceptance probability is the stage 14 termination probability, i.e., approximately 55%.

- **Assume Defect Quantity Equals 16**

If 16 of the 310 procedures are defective, then the percentage of defectives is

```
16/310 // N
```

```
0.0516129
```

and the percentage of non-defectives is

```
1 - %
```

```
0.948387
```

The initial quantity of defective procedures is assigned as the value of the symbol *initdef*:

```
initdef = 16;
```

The pseudorandom number generating function is:

```
fun[{stage_Integer, cumdef_Integer}] :=
 {stage + 1, cumdef + Random[HypergeometricDistribution[
     sample[[stage+1]], initdef - cumdef, proc[[stage+1]]]]}
```

The function that will test the simulation against the acceptance rules is:

```
testfun[{stage_Integer, cumdef_Integer}] := accept[stage] < cumdef
```

A simulation of 25,000 hypergeometric tests is generated as follows:

```
simlist = Table[
   Length[NestWhileList[fun, {0, 0}, testfun, 1, Length[sample]]] - 1,
   {simqty}];
```

The symbol *simlist* contains a list of the stages that the simulations ended at. The quantity of terminations at each stage are assigned as the value of *simlist16*:

```
simlist16 =
  Table[Length[Select[simlist, #1 == i &]], {i, 1, Length[sample]}];
```

The simulation results are appended to the simulation file so that they can be retrieved for subsequent analysis if need be.

```
Save[simfile, simlist16]
```

These simulation results may be easily retrieved thus:

```
<< "hypersimfile3";
```

Stage-by-stage termination probabilities are:

```
TableForm[Transpose[{Range[1, 15], N[simlist16/simqty]}],
 TableHeadings → {None, {"Stage", "Termination Probability"}},
 TableAlignments → Center]
```

Stage	Termination Probability
1	0.33352
2	0.13024
3	0.07236
4	0.0516
5	0.0366
6	0.03036
7	0.02564
8	0.02008
9	0.01772
10	0.0154
11	0.0142
12	0.01332
13	0.0122
14	0.00956
15	0.2172

The terminations at stages 1 - 14 are acceptances. The terminations at the last stage are simulations that went the distance. Cumulative termination probabilities are:

```
TableForm[Transpose[{Range[0, 15], N[FoldList[Plus, 0, simlist16/simqty]]}],
 TableHeadings →
  {None, {"Stage", "Cumulative Termination Probability"}},
 TableAlignments → Center]
```

Stage	Cumulative Termination Probability
0	0.
1	0.33352
2	0.46376
3	0.53612
4	0.58772
5	0.62432
6	0.65468
7	0.68032
8	0.7004
9	0.71812
10	0.73352
11	0.74772
12	0.76104
13	0.77324
14	0.7828
15	1.

The stage 1 acceptance probability is:

```
PDF[HypergeometricDistribution[20, 16, 310], 0] // N
```

```
0.334635
```

The stage 1 acceptance probability obtained by simulation is consistent with the numerical result above. The final cumulative acceptance probability is the stage 14 termination probability, i.e., approximately 78%.

- **Assume Defect Quantity Equals 13**

If 13 of the 310 procedures are defective, then the percentage of defectives is

```
13/310 // N
```

```
0.0419355
```

and the percentage of non-defectives is

```
1 - %
```

```
0.958065
```

The initial quantity of defective procedures is assigned as the value of the symbol *initdef*:

```
initdef = 13;
```

The pseudorandom number generating function is:

```
fun[{stage_Integer, cumdef_Integer}] :=
 {stage + 1, cumdef + Random[HypergeometricDistribution[
     sample[[stage+1]], initdef - cumdef, proc[[stage+1]]]]}
```

The function that will test the simulation against the acceptance rules is:

```
testfun[{stage_Integer, cumdef_Integer}] := accept[stage] < cumdef
```

A simulation of 25,000 hypergeometric tests is generated as follows:

```
simlist = Table[
   Length[NestWhileList[fun, {0, 0}, testfun, 1, Length[sample]]] - 1,
   {simqty}];
```

The symbol *simlist* contains a list of the stages that the simulations ended at. The quantity of terminations at each stage are assigned as the value of *simlist13*:

```
simlist13 =
  Table[Length[Select[simlist, #1 == i &]], {i, 1, Length[sample]}];
```

The simulation results are appended to the simulation file so that they can be retrieved for subsequent analysis if need be.

```
Save[simfile, simlist13]
```

These simulation results may be easily retrieved thus:

```
<< "hypersimfile3";
```

The terminations at stages 1 - 14 are acceptances. The terminations at the last stage are simulations that went the distance. Cumulative termination probabilities are:

```
TableForm[Transpose[{Range[1, 15], N[simlist13/simqty]}],
 TableHeadings → {None, {"Stage", "Termination Probability"}},
 TableAlignments → Center]
```

Stage	Termination Probability
1	0.41144
2	0.16356
3	0.09296
4	0.063
5	0.04632
6	0.03588
7	0.03156
8	0.02708
9	0.02508
10	0.02268
11	0.02148
12	0.01988
13	0.02124
14	0.01784
15	0.

Cumulative termination probabilities are:

```
TableForm[Transpose[{Range[0, 15], N[FoldList[Plus, 0, simlist13/simqty]]}],
 TableHeadings →
  {None, {"Stage", "Cumulative Termination Probability"}},
 TableAlignments → Center]
```

Stage	Cumulative Termination Probability
0	0.
1	0.41144
2	0.575
3	0.66796
4	0.73096
5	0.77728
6	0.81316
7	0.84472
8	0.8718
9	0.89688
10	0.91956
11	0.94104
12	0.96092
13	0.98216
14	1.
15	1.

The stage 1 acceptance probability is:

```
PDF[HypergeometricDistribution[20, 13, 310], 0] // N
```

```
0.412781
```

The stage 1 acceptance probability obtained by simulation is consistent with the numerical result above. The final cumulative acceptance probability is the stage 14 termination probability, i.e., approximately 100%.

- **Assume Defect Quantity Equals 10**

If 10 of the 310 procedures are defective, then the percentage of defectives is

```
10/310 // N
```

```
0.0322581
```

and the percentage of non-defectives is

```
1 - %
```

```
0.967742
```

The initial quantity of defective procedures is assigned as the value of the symbol *initdef*:

```
initdef = 10;
```

The pseudorandom number generating function is:

```
fun[{stage_Integer, cumdef_Integer}] :=
 {stage + 1, cumdef + Random[HypergeometricDistribution[
     sample[[stage+1]], initdef - cumdef, proc[[stage+1]]]]}
```

The function that will test the simulation against the acceptance rules is:

```
testfun[{stage_Integer, cumdef_Integer}] := accept[stage] < cumdef
```

A simulation of 25,000 hypergeometric tests is generated as follows:

```
simlist = Table[
   Length[NestWhileList[fun, {0, 0}, testfun, 1, Length[sample]]] - 1,
   {simqty}];
```

The symbol *simlist* contains a list of the stages that the simulations ended at. The quantity of terminations at each stage are assigned as the value of *simlist10*:

```
simlist10 =
  Table[Length[Select[simlist, #1 == i &]], {i, 1, Length[sample]}];
```

The simulation results are appended to the simulation file so that they can be retrieved for subsequent analysis if need be.

```
Save[simfile, simlist10]
```

These simulation results may be easily retrieved thus:

```
<< "hypersimfile3";
```

The terminations at stages 1 - 14 are acceptances. The terminations at the last stage are simulations that went the distance. Cumulative termination probabilities are:

```
TableForm[Transpose[{Range[1, 15], N[simlist10/simqty]}],
 TableHeadings → {None, {"Stage", "Termination Probability"}},
 TableAlignments → Center]
```

Stage	Termination Probability
1	0.50552
2	0.19548
3	0.1014
4	0.06412
5	0.04216
6	0.03136
7	0.02356
8	0.01624
9	0.01084
10	0.00672
11	0.0026
12	0.
13	0.
14	0.
15	0.

Cumulative termination probabilities are:

```
TableForm[Transpose[{Range[0, 15], N[FoldList[Plus, 0, simlist10/simqty]]}],
 TableHeadings →
  {None, {"Stage", "Cumulative Termination Probability"}},
 TableAlignments → Center]
```

Stage	Cumulative Termination Probability
0	0.
1	0.50552
2	0.701
3	0.8024
4	0.86652
5	0.90868
6	0.94004
7	0.9636
8	0.97984
9	0.99068
10	0.9974
11	1.
12	1.
13	1.
14	1.
15	1.

The stage 1 acceptance probability is:

```
PDF[HypergeometricDistribution[20, 10, 310], 0] // N
```

```
0.508067
```

The stage 1 acceptance probability obtained by simulation is consistent with the numerical result above. The final cumulative acceptance probability is the stage 14 termination probability, i.e., approximately 100%.

- **Assume Defect Quantity Equals 7**

If 7 of the 310 procedures are defective, then the percentage of defectives is

$$\frac{7}{310} \text{ // N}$$

```
0.0225806
```

and the percentage of non-defectives is

```
1 - %
```

```
0.977419
```

The initial quantity of defective procedures is assigned as the value of the symbol *initdef*:

```
initdef = 7;
```

The pseudorandom number generating function is:

```
fun[{stage_Integer, cumdef_Integer}] :=
 {stage + 1, cumdef + Random[HypergeometricDistribution[
     sample[[stage+1]], initdef - cumdef, proc[[stage+1]]]]}
```

The function that will test the simulation against the acceptance rules is:

```
testfun[{stage_Integer, cumdef_Integer}] := accept[stage] < cumdef
```

A simulation of 25,000 hypergeometric tests is generated as follows:

```
simlist = Table[
   Length[NestWhileList[fun, {0, 0}, testfun, 1, Length[sample]]] - 1,
   {simqty}];
```

The symbol *simlist* contains a list of the stages that the simulations ended at. The quantity of terminations at each stage are assigned as the value of *simlist7*:

```
simlist7 =
  Table[Length[Select[simlist, #1 == i &]], {i, 1, Length[sample]}];
```

The simulation results are appended to the simulation file so that they can be retrieved for subsequent analysis if need be.

```
Save[simfile, simlist7]
```

These simulation results may be easily retrieved thus:

```
<< "hypersimfile3";
```

The terminations at stages 1 - 14 are acceptances. The terminations at the last stage are simulations that went the distance. Cumulative termination probabilities are:

```
TableForm[Transpose[{Range[1, 15], N[simlist7/simqty]}],
 TableHeadings → {None, {"Stage", "Termination Probability"}},
 TableAlignments → Center]
```

Stage	Termination Probability
1	0.6214
2	0.20276
3	0.08992
4	0.04584
5	0.02376
6	0.01072
7	0.0046
8	0.001
9	0.
10	0.
11	0.
12	0.
13	0.
14	0.
15	0.

Cumulative termination probabilities are:

```
TableForm[Transpose[{Range[0, 15], N[FoldList[Plus, 0, simlist7/simqty]]}],
 TableHeadings →
  {None, {"Stage", "Cumulative Termination Probability"}},
 TableAlignments → Center]
```

Stage	Cumulative Termination Probability
0	0.
1	0.6214
2	0.82416
3	0.91408
4	0.95992
5	0.98368
6	0.9944
7	0.999
8	1.
9	1.
10	1.
11	1.
12	1.
13	1.
14	1.
15	1.

The stage 1 acceptance probability is:

```
PDF[HypergeometricDistribution[20, 7, 310], 0] // N
```

```
0.624015
```

The stage 1 acceptance probability obtained by simulation is consistent with the numerical result above. The final cumulative acceptance probability is the stage 14 termination probability, i.e., approximately 100%.

- **Assume Defect Quantity Equals 4**

If 4 of the 310 procedures are defective, then the percentage of defectives is

```
4/310 // N
```

```
0.0129032
```

and the percentage of non-defectives is

```
1 - %
```

```
0.987097
```

The initial quantity of defective procedures is assigned as the value of the symbol *initdef*:

```
initdef = 4;
```

The pseudorandom number generating function is:

```
fun[{stage_Integer, cumdef_Integer}] :=
 {stage + 1, cumdef + Random[HypergeometricDistribution[
     sample[[stage+1]], initdef - cumdef, proc[[stage+1]]]]}
```

The function that will test the simulation against the acceptance rules is:

```
testfun[{stage_Integer, cumdef_Integer}] := accept[stage] < cumdef
```

A simulation of 25,000 hypergeometric tests is generated as follows:

```
simlist = Table[
   Length[NestWhileList[fun, {0, 0}, testfun, 1, Length[sample]]] - 1,
   {simqty}];
```

The symbol *simlist* contains a list of the stages that the simulations ended at. The quantity of terminations at each stage are assigned as the value of *simlist4*:

```
simlist4 =
  Table[Length[Select[simlist, #1 = i &]], {i, 1, Length[sample]}];
```

The simulation results are appended to the simulation file so that they can be retrieved for subsequent analysis if need be.

```
Save[simfile, simlist4]
```

These simulation results may be easily retrieved thus:

```
<< "hypersimfile3";
```

The terminations at stages 1 - 14 are acceptances. The terminations at the last stage are simulations that went the distance. Cumulative termination probabilities are:

```
TableForm[Transpose[{Range[1, 15], N[simlist4/simqty]}],
 TableHeadings → {None, {"Stage", "Termination Probability"}},
 TableAlignments → Center]
```

Stage	Termination Probability
1	0.7682
2	0.16852
3	0.04868
4	0.01292
5	0.00168
6	0.
7	0.
8	0.
9	0.
10	0.
11	0.
12	0.
13	0.
14	0.
15	0.

Cumulative termination probabilities are:

```
TableForm[Transpose[{Range[0, 15], N[FoldList[Plus, 0, simlist4/simqty]]}],
 TableHeadings →
  {None, {"Stage", "Cumulative Termination Probability"}},
 TableAlignments → Center]
```

Stage	Cumulative Termination Probability
0	0.
1	0.7682
2	0.93672
3	0.9854
4	0.99832
5	1.
6	1.
7	1.
8	1.
9	1.
10	1.
11	1.
12	1.
13	1.
14	1.
15	1.

The stage 1 acceptance probability is:

```
PDF[HypergeometricDistribution[20, 4, 310], 0] // N

0.764823
```

The stage 1 acceptance probability obtained by simulation is consistent with the numerical result above. The final cumulative acceptance probability is the stage 14 termination probability, i.e., approximately 100%.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix F

Distribution List

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix F

Distribution List

No. of Copies	Organization
7	Defense Technical Information Center ATTN: DTIC-OMI 8725 John J. Kingman Road, Suite 0944 Fort Belvoir, VA 22060-6218
15	Director US Army Materiel Systems Analysis Activity ATTN: AMXSY- A (M. Cushing, D. Mortin) AMXSY-DDS (3 cys) 392 Hopkins Road Aberdeen Proving Ground, MD 21009-5071

THIS PAGE INTENTIONALLY LEFT BLANK